

---

# MPC8533E PowerQUICC™ III Integrated Host Processor Family Reference Manual

**Supports**  
MPC8533E  
MPC8533

MPC8533ERM  
Rev. 1  
10/2007



## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **Web Support:**

<http://www.freescale.com/support>

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
+1-800-521-6274 or  
+1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku  
Tokyo 153-0064  
Japan  
0120 191014 or  
+81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### **For Literature Requests Only:**

Freescale Semiconductor  
Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
+1-800 441-2447 or  
+1-303-675-2140  
Fax: +1-303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. The PowerPC name is a trademark of IBM Corp. and is used under license. IEEE 802.1, 802.2, 802.3, 802.11, and 1149.1 are registered trademarks of the Institute of Electrical and Electronics Engineers, Inc. (IEEE). This product is not endorsed or approved by the IEEE. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc., 2007. Printed in the United States of America. All rights reserved.



<b>Part I—Overview</b>	<b>I</b>
Overview	<b>1</b>
Memory Map	<b>2</b>
Signal Descriptions	<b>3</b>
Reset, Clocking, and Initialization	<b>4</b>
<b>Part II—e500 Core Complex and L2 Cache</b>	<b>II</b>
Core Complex Overview	<b>5</b>
Core Register Summary	<b>6</b>
L2 Look-Aside Cache/SRAM	<b>7</b>
<b>Part III—Memory, Security, and I/O Interfaces</b>	<b>III</b>
e500 Coherency Module	<b>8</b>
DDR Memory Controller	<b>9</b>
Programmable Interrupt Controller	<b>10</b>
I <sup>2</sup> C Interfaces	<b>11</b>
Security Engine (SEC) 2.1	<b>12</b>
DUART	<b>13</b>
Local Bus Controller	<b>14</b>
Enhanced Three-Speed Ethernet Controllers	<b>15</b>
DMA Controller	<b>16</b>
PCI Bus Interface	<b>17</b>
PCI Express Interface Controller	<b>18</b>
<b>Part IV—Global Functions and Debug</b>	<b>IV</b>
Global Utilities	<b>19</b>
Device Performance Monitor	<b>20</b>
Debug Features and Watchpoint Facility	<b>21</b>
Revision History	<b>A</b>
Complete List of Configuration, Control, and Status Registers	<b>B</b>
Glossary	<b>GLO</b>
Index	<b>IND</b>

<b>I</b>	<b>Part I—Overview</b>
<b>1</b>	Overview
<b>2</b>	Memory Map
<b>3</b>	Signal Descriptions
<b>4</b>	Reset, Clocking, and Initialization
<b>II</b>	<b>Part II—e500 Core Complex and L2 Cache</b>
<b>5</b>	Core Complex Overview
<b>6</b>	Core Register Summary
<b>7</b>	L2 Look-Aside Cache/SRAM
<b>III</b>	<b>Part III—Memory, Security, and I/O Interfaces</b>
<b>8</b>	e500 Coherency Module
<b>9</b>	DDR Memory Controller
<b>10</b>	Programmable Interrupt Controller
<b>11</b>	I <sup>2</sup> C Interfaces
<b>12</b>	Security Engine (SEC) 2.1
<b>13</b>	DUART
<b>14</b>	Local Bus Controller
<b>15</b>	Enhanced Three-Speed Ethernet Controllers
<b>16</b>	DMA Controller
<b>17</b>	PCI Bus Interface
<b>18</b>	PCI Express Interface Controller
<b>IV</b>	<b>Part IV—Global Functions and Debug</b>
<b>19</b>	Global Utilities
<b>20</b>	Device Performance Monitor
<b>21</b>	Debug Features and Watchpoint Facility
<b>A</b>	Revision History
<b>B</b>	Complete List of Configuration, Control, and Status Registers
<b>GLO</b>	Glossary
<b>IND</b>	Index



# Contents

Paragraph Number	Title	Page Number
<b>About This Book</b>		
<b>Part I</b>		
<b>Overview</b>		
<b>Chapter 1</b>		
<b>Overview</b>		
1.1	Introduction.....	1-1
1.2	MPC8533E Overview.....	1-2
1.2.1	Key Features .....	1-2
1.3	MPC8533E Architecture Overview.....	1-8
1.3.1	e500 Core Overview .....	1-8
1.3.2	On-Chip Memory Unit.....	1-12
1.3.3	On-Chip Memory as Memory-Mapped SRAM.....	1-13
1.3.4	On-Chip Memory as L2 Cache.....	1-13
1.3.5	e500 Coherency Module (ECM).....	1-14
1.3.6	DDR SDRAM Controller .....	1-14
1.3.7	Programmable Interrupt Controller (PIC).....	1-15
1.3.8	Integrated Security Engine (SEC) for the MPC8533E .....	1-15
1.3.9	I <sup>2</sup> C Controllers.....	1-16
1.3.10	Boot Sequencer.....	1-17
1.3.11	Dual Universal Asynchronous Receiver/Transmitter (DUART).....	1-17
1.3.12	Local Bus Controller.....	1-17
1.3.13	Enhanced Three-Speed Ethernet Controllers (eTSECs).....	1-18
1.3.14	Ocean Switch Fabric .....	1-19
1.3.15	Integrated DMA.....	1-19
1.3.16	PCI Controller.....	1-19
1.3.17	PCI Express Interface .....	1-19
1.3.18	Power Management .....	1-20
1.3.19	Clocking.....	1-20
1.3.20	Address Map.....	1-20
1.3.21	Processing Across the On-Chip Fabric.....	1-21
1.3.22	Data Processing with the e500 Coherency Module .....	1-21
1.4	Application Examples.....	1-21
1.4.1	Multifunction Router Application .....	1-22
1.4.2	Multifunction Printer Application .....	1-23
1.4.3	Security Appliance.....	1-24
1.4.4	IP SAN Host Adapter .....	1-25
1.4.5	VoIP Aggregation Application.....	1-26

# Contents

Paragraph Number	Title	Page Number
<b>Chapter 2</b>		
<b>Memory Map</b>		
2.1	Local Memory Map Overview and Example .....	2-1
2.2	Address Translation and Mapping .....	2-3
2.2.1	SRAM Windows .....	2-4
2.2.2	Window into Configuration Space.....	2-4
2.2.3	Local Access Windows.....	2-4
2.2.3.1	Local Access Register Memory Map .....	2-5
2.2.3.2	Local Access IP Block Revision Register 1 (LAIPBRR1).....	2-6
2.2.3.3	Local Access IP Block Revision Register 2 (LAIPBRR2).....	2-6
2.2.3.4	Local Access Window <i>n</i> Base Address Registers (LAWBAR0–LAWBAR9).....	2-7
2.2.3.5	Local Access Window <i>n</i> Attributes Registers (LAWAR0–LAWAR9).....	2-7
2.2.3.6	Precedence of Local Access Windows .....	2-8
2.2.3.7	Configuring Local Access Windows .....	2-8
2.2.3.8	Distinguishing Local Access Windows from Other Mapping Functions .....	2-9
2.2.3.9	Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects.....	2-9
2.2.4	Outbound Address Translation and Mapping Windows.....	2-9
2.2.5	Inbound Address Translation and Mapping Windows .....	2-9
2.2.5.1	PCI Inbound ATMU .....	2-10
2.2.5.2	PCI Express Inbound ATMU .....	2-10
2.2.5.3	Illegal Interaction Between Inbound ATMUs and Local Access Windows .....	2-10
2.3	Configuration, Control, and Status Register Map.....	2-10
2.3.1	Accessing CCSR Memory from the Local Processor.....	2-10
2.3.2	Accessing CCSR Memory from External Masters .....	2-11
2.3.3	Organization of CCSR Memory .....	2-11
2.3.4	General Utilities Registers .....	2-12
2.3.5	Interrupt Controller and CCSR.....	2-13
2.3.6	Device-Specific Utilities.....	2-13
2.4	Complete CCSR Map .....	2-14

## Chapter 3 Signal Descriptions

3.1	Signals Overview .....	3-1
3.2	Configuration Signals Sampled at Reset .....	3-15
3.3	Output Signal States During Reset .....	3-16

## Chapter 4 Reset, Clocking, and Initialization

# Contents

Paragraph Number	Title	Page Number
4.1	Overview.....	4-1
4.2	External Signal Descriptions .....	4-1
4.2.1	System Control Signals.....	4-2
4.2.2	Clock Signals .....	4-3
4.3	Memory Map/Register Definition .....	4-4
4.3.1	Local Configuration Control.....	4-4
4.3.1.1	Accessing Configuration, Control, and Status Registers.....	4-4
4.3.1.1.1	Updating CCSRBAR .....	4-4
4.3.1.1.2	Configuration, Control, and Status Base Address Register (CCSRBAR).....	4-5
4.3.1.2	Accessing Alternate Configuration Space .....	4-6
4.3.1.2.1	Alternate Configuration Base Address Register (ALTCBAR).....	4-6
4.3.1.2.2	Alternate Configuration Attribute Register (ALTCAR).....	4-6
4.3.1.3	Boot Page Translation.....	4-7
4.3.1.3.1	Boot Page Translation Register (BPTR).....	4-7
4.3.2	Boot Sequencer.....	4-8
4.4	Functional Description.....	4-8
4.4.1	Reset Operations .....	4-8
4.4.1.1	Soft Reset.....	4-8
4.4.1.2	Hard Reset .....	4-8
4.4.2	Power-On Reset Sequence.....	4-9
4.4.3	Power-On Reset Configuration.....	4-10
4.4.3.1	System PLL Ratio.....	4-11
4.4.3.2	e500 Core PLL Ratio .....	4-12
4.4.3.3	SEC Frequency Ratio Configuration .....	4-13
4.4.3.4	Boot ROM Location .....	4-13
4.4.3.5	Host/Agent Configuration .....	4-14
4.4.3.6	I/O Port Selection .....	4-14
4.4.3.7	CPU Boot Configuration .....	4-15
4.4.3.8	Boot Sequencer Configuration .....	4-16
4.4.3.9	DDR SDRAM Type.....	4-17
4.4.3.10	eTSEC1 Width.....	4-17
4.4.3.11	eTSEC3 Width.....	4-17
4.4.3.12	eTSEC1 Protocol .....	4-18
4.4.3.13	eTSEC3 Protocol .....	4-18
4.4.3.14	PCI Clock Selection.....	4-19
4.4.3.15	PCI Speed Configuration.....	4-19
4.4.3.16	PCI I/O Impedance .....	4-20
4.4.3.17	PCI Arbiter Configuration .....	4-20
4.4.3.18	Memory Debug Configuration .....	4-20
4.4.3.19	DDR Debug Configuration.....	4-20
4.4.3.20	General-Purpose POR Configuration .....	4-21

# Contents

Paragraph Number	Title	Page Number
4.4.4	Clocking.....	4-21
4.4.4.1	System Clock/PCI Clock .....	4-21
4.4.4.2	PCI Express .....	4-22
4.4.4.2.1	Minimum Frequency Requirements .....	4-22
4.4.4.3	Ethernet Clocks.....	4-23
4.4.4.4	Real Time Clock.....	4-23

## Part II e500 Core Complex and L2 Cache

### Chapter 5 Core Complex Overview

5.1	Overview.....	5-1
5.1.1	Upward Compatibility .....	5-3
5.1.2	Core Complex Summary .....	5-3
5.2	e500 Processor and System Version Numbers.....	5-4
5.3	Features.....	5-5
5.3.1	e500v2 Differences .....	5-10
5.4	Instruction Set .....	5-11
5.5	Instruction Flow .....	5-13
5.5.1	Initial Instruction Fetch.....	5-13
5.5.2	Branch Detection and Prediction .....	5-13
5.5.3	e500 Execution Pipeline .....	5-14
5.6	Programming Model .....	5-16
5.7	On-Chip Cache Implementation .....	5-18
5.8	Interrupts and Exception Handling.....	5-18
5.8.1	Exception Handling .....	5-18
5.8.2	Interrupt Classes .....	5-19
5.8.3	Interrupt Types .....	5-19
5.8.4	Upper Bound on Interrupt Latencies .....	5-20
5.8.5	Interrupt Registers.....	5-20
5.9	Memory Management.....	5-22
5.9.1	Address Translation .....	5-24
5.9.2	MMU Assist Registers (MAS0–MAS4 and MAS6–MAS7).....	5-25
5.9.3	Process ID Registers (PID0–PID2).....	5-26
5.9.4	TLB Coherency.....	5-26
5.10	Memory Coherency .....	5-26
5.10.1	Atomic Update Memory References .....	5-27
5.10.2	Memory Access Ordering.....	5-27
5.10.3	Cache Control Instructions .....	5-27

# Contents

Paragraph Number	Title	Page Number
5.10.4	Programmable Page Characteristics .....	5-27
5.11	Core Complex Bus (CCB) .....	5-27
5.12	Performance Monitoring .....	5-28
5.12.1	Global Control Register .....	5-28
5.12.2	Performance Monitor Counter Registers .....	5-28
5.12.3	Local Control Registers .....	5-29
5.13	Legacy Support of Power Architecture Technology .....	5-29
5.13.1	Instruction Set Compatibility .....	5-29
5.13.1.1	User Instruction Set .....	5-29
5.13.1.2	Supervisor Instruction Set .....	5-30
5.13.2	Memory Subsystem .....	5-30
5.13.3	Exception Handling .....	5-30
5.13.4	Memory Management .....	5-30
5.13.5	Reset .....	5-30
5.13.6	Little-Endian Mode .....	5-31
5.14	PowerQUICC III Implementation Details .....	5-31

## Chapter 6 Core Register Summary

6.1	Overview .....	6-1
6.1.1	Register Set .....	6-1
6.2	Register Model for 32-Bit Implementations .....	6-3
6.2.1	Special-Purpose Registers (SPRs) .....	6-4
6.3	Registers for Computational Operations .....	6-8
6.3.1	General-Purpose Registers (GPRs) .....	6-8
6.3.2	Integer Exception Register (XER) .....	6-8
6.4	Registers for Branch Operations .....	6-9
6.4.1	Condition Register (CR) .....	6-9
6.4.2	Link Register (LR) .....	6-11
6.4.3	Count Register (CTR) .....	6-11
6.5	Processor Control Registers .....	6-11
6.5.1	Machine State Register (MSR) .....	6-11
6.5.2	Processor ID Register (PIR) .....	6-13
6.5.3	Processor Version Register (PVR) .....	6-13
6.5.4	System Version Register (SVR) .....	6-14
6.6	Timer Registers .....	6-14
6.6.1	Timer Control Register (TCR) .....	6-14
6.6.2	Timer Status Register (TSR) .....	6-15
6.6.3	Time Base Registers .....	6-16
6.6.4	Decrementer Register .....	6-16

# Contents

Paragraph Number	Title	Page Number
6.6.5	Decrementer Auto-Reload Register (DECAR).....	6-17
6.7	Interrupt Registers.....	6-17
6.7.1	Interrupt Registers Defined by the Embedded and Base Categories .....	6-17
6.7.1.1	Save/Restore Register 0 (SRR0).....	6-17
6.7.1.2	Save/Restore Register 1 (SRR1).....	6-17
6.7.1.3	Critical Save/Restore Register 0 (CSRR0).....	6-17
6.7.1.4	Critical Save/Restore Register 1 (CSRR1).....	6-18
6.7.1.5	Data Exception Address Register (DEAR).....	6-18
6.7.1.6	Interrupt Vector Prefix Register (IVPR).....	6-18
6.7.1.7	Interrupt Vector Offset Registers (IVOR <sub>n</sub> ).....	6-18
6.7.1.8	Exception Syndrome Register (ESR) .....	6-19
6.7.2	Additional Interrupt Registers .....	6-20
6.7.2.1	Machine Check Save/Restore Register 0 (MCSRR0) .....	6-20
6.7.2.2	Machine Check Save/Restore Register 1 (MCSRR1) .....	6-20
6.7.2.3	Machine Check Address Register (MCAR/MCARU) .....	6-21
6.7.2.4	Machine Check Syndrome Register (MCSR).....	6-21
6.8	Software-Use SPRs (SPRG0–SPRG7 and USPRG0) .....	6-22
6.9	Branch Target Buffer (BTB) Registers .....	6-23
6.9.1	Branch Buffer Entry Address Register (BBEAR) .....	6-23
6.9.2	Branch Buffer Target Address Register (BBTAR) .....	6-23
6.9.3	Branch Unit Control and Status Register (BUCSR).....	6-24
6.10	Hardware Implementation-Dependent Registers .....	6-25
6.10.1	Hardware Implementation-Dependent Register 0 (HID0).....	6-25
6.10.2	Hardware Implementation-Dependent Register 1 (HID1).....	6-26
6.11	L1 Cache Configuration Registers.....	6-28
6.11.1	L1 Cache Control and Status Register 0 (L1CSR0) .....	6-28
6.11.2	L1 Cache Control and Status Register 1 (L1CSR1) .....	6-29
6.11.3	L1 Cache Configuration Register 0 (L1CFG0) .....	6-30
6.11.4	L1 Cache Configuration Register 1 (L1CFG1) .....	6-31
6.12	MMU Registers.....	6-32
6.12.1	Process ID Registers (PID0–PID2).....	6-32
6.12.2	MMU Control and Status Register 0 (MMUCSR0) .....	6-32
6.12.3	MMU Configuration Register (MMUCFG) .....	6-32
6.12.4	TLB Configuration Registers (TLB <sub>n</sub> CFG).....	6-33
6.12.4.1	TLB0 Configuration Register 0 (TLB0CFG).....	6-33
6.12.4.2	TLB1 Configuration Register 1 (TLB1CFG).....	6-34
6.12.5	MMU Assist Registers.....	6-35
6.12.5.1	MAS Register 0 (MAS0) .....	6-35
6.12.5.2	MAS Register 1 (MAS1) .....	6-35
6.12.5.3	MAS Register 2 (MAS2).....	6-36
6.12.5.4	MAS Register 3 (MAS3).....	6-37

# Contents

Paragraph Number	Title	Page Number
6.12.5.5	MAS Register 4 (MAS4) .....	6-38
6.12.5.6	MAS Register 6 (MAS6) .....	6-38
6.12.5.7	MAS Register 7 (MAS7) .....	6-39
6.13	Debug Registers .....	6-39
6.13.1	Debug Control Registers (DBCR0–DBCR2) .....	6-39
6.13.1.1	Debug Control Register 0 (DBCR0).....	6-39
6.13.1.2	Debug Control Register 1 (DBCR1).....	6-41
6.13.1.3	Debug Control Register 2 (DBCR2).....	6-42
6.13.2	Debug Status Register (DBSR).....	6-43
6.13.3	Instruction Address Compare Registers (IAC1–IAC2) .....	6-45
6.13.4	Data Address Compare Registers (DAC1–DAC2).....	6-45
6.14	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR).....	6-45
6.14.1	Accumulator (ACC).....	6-47
6.15	Performance Monitor Registers (PMRs) .....	6-48
6.15.1	Global Control Register 0 (PMGC0, UPMGC0).....	6-49
6.15.2	Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3) .....	6-50
6.15.3	Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3) .....	6-51
6.15.4	Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3).....	6-52

## Chapter 7 L2 Look-Aside Cache/SRAM

7.1	L2 Cache Overview .....	7-1
7.1.1	L2 Cache and SRAM Features .....	7-2
7.2	L2 Cache and SRAM Organization .....	7-4
7.2.1	Accessing the On-Chip Array as an L2 Cache .....	7-5
7.2.2	Accessing the On-Chip Array as an SRAM .....	7-5
7.2.3	Connection of the On-Chip Memory to the System .....	7-7
7.3	Memory Map/Register Definition .....	7-8
7.3.1	L2/SRAM Register Descriptions .....	7-10
7.3.1.1	L2 Control Register (L2CTL).....	7-10
7.3.1.2	L2 Cache External Write Registers .....	7-13
7.3.1.2.1	L2 Cache External Write Address Registers 0–3 (L2CEWAR <sub>n</sub> ) .....	7-13
7.3.1.2.2	L2 Cache External Write Address Registers Extended Address 0–3 (L2CEWAREA <sub>n</sub> ).....	7-14
7.3.1.2.3	L2 Cache External Write Control Registers 0–3 (L2CEWCR <sub>n</sub> ).....	7-14
7.3.1.3	L2 Memory-Mapped SRAM Registers .....	7-15
7.3.1.3.1	L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR <sub>n</sub> ) .....	7-16
7.3.1.3.2	L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA <sub>n</sub> ).....	7-17

# Contents

Paragraph Number	Title	Page Number
7.3.1.4	L2 Error Registers.....	7-17
7.3.1.4.1	Error Injection Registers.....	7-18
7.3.1.4.2	Error Control and Capture Registers .....	7-20
7.4	External Writes to the L2 Cache (Cache Stashing).....	7-25
7.4.1	Stash-Only Cache Regions .....	7-26
7.5	L2 Cache Timing .....	7-27
7.6	L2 Cache and SRAM Coherency.....	7-27
7.6.1	L2 Cache Coherency Rules.....	7-28
7.6.2	Memory-Mapped SRAM Coherency Rules .....	7-29
7.7	L2 Cache Locking.....	7-29
7.7.1	Locking the Entire L2 Cache .....	7-29
7.7.2	Locking Programmed Memory Ranges.....	7-30
7.7.3	Locking Selected Lines.....	7-30
7.7.4	Clearing Locks on Selected Lines .....	7-30
7.7.5	Flash Clearing of Instruction and Data Locks .....	7-31
7.7.6	Locks with Stale Data .....	7-31
7.8	PLRU L2 Replacement Policy.....	7-31
7.8.1	PLRU Bit Update Considerations.....	7-32
7.8.2	Allocation of Lines .....	7-32
7.9	L2 Cache Operation .....	7-33
7.9.1	Initialization .....	7-33
7.9.1.1	L2 Cache Initialization .....	7-33
7.9.1.2	Memory-Mapped SRAM Initialization .....	7-33
7.9.2	Flash Invalidation of the L2 Cache.....	7-34
7.9.3	Managing Errors .....	7-34
7.9.3.1	ECC Errors.....	7-34
7.9.3.2	Tag Parity Errors.....	7-34
7.9.4	L2 Cache States .....	7-34
7.9.5	L2 State Transitions .....	7-35
7.9.6	Error Checking and Correcting (ECC) .....	7-39

## Part III

### Memory, Security, and I/O Interfaces

#### Chapter 8

#### e500 Coherency Module

8.1	Introduction.....	8-1
8.1.1	Overview.....	8-2
8.1.2	Features.....	8-2
8.2	Memory Map/Register Definition .....	8-3



# Contents

Paragraph Number	Title	Page Number
8.2.1	Register Descriptions .....	8-3
8.2.1.1	ECM CCB Address Configuration Register (EEBACR) .....	8-3
8.2.1.2	ECM CCB Port Configuration Register (EEBPCR) .....	8-4
8.2.1.3	ECM IP Block Revision Register 1 (EIPBRR1) .....	8-5
8.2.1.4	ECM IP Block Revision Register 2 (EIPBRR2) .....	8-5
8.2.1.5	ECM Error Detect Register (EEDR) .....	8-6
8.2.1.6	ECM Error Enable Register (EEER) .....	8-7
8.2.1.7	ECM Error Attributes Capture Register (EEATR) .....	8-7
8.2.1.8	ECM Error Low Address Capture Register (EELADR) .....	8-8
8.2.1.9	ECM Error High Address Capture Register (EEHADR) .....	8-9
8.3	Functional Description .....	8-9
8.3.1	I/O Arbiter .....	8-9
8.3.2	CCB Arbiter .....	8-9
8.3.3	Transaction Queue .....	8-10
8.3.4	Global Data Multiplexor .....	8-10
8.3.5	CCB Interface .....	8-10
8.4	Initialization/Application Information .....	8-10

## Chapter 9 DDR Memory Controller

9.1	Introduction .....	9-1
9.2	Features .....	9-2
9.2.1	Modes of Operation .....	9-3
9.3	External Signal Descriptions .....	9-3
9.3.1	Signals Overview .....	9-3
9.3.2	Detailed Signal Descriptions .....	9-5
9.3.2.1	Memory Interface Signals .....	9-5
9.3.2.2	Clock Interface Signals .....	9-9
9.3.2.3	Debug Signals .....	9-9
9.4	Memory Map/Register Definition .....	9-9
9.4.1	Register Descriptions .....	9-11
9.4.1.1	Chip Select Memory Bounds (CS <sub>n</sub> _BNDS) .....	9-11
9.4.1.2	Chip Select Configuration (CS <sub>n</sub> _CONFIG) .....	9-11
9.4.1.3	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3) .....	9-13
9.4.1.4	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0) .....	9-14
9.4.1.5	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1) .....	9-16
9.4.1.6	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2) .....	9-18
9.4.1.7	DDR SDRAM Control Configuration (DDR_SDRAM_CFG) .....	9-20
9.4.1.8	DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2) .....	9-23
9.4.1.9	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE) .....	9-25

# Contents

Paragraph Number	Title	Page Number
9.4.1.10	DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2).....	9-26
9.4.1.11	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL).....	9-26
9.4.1.12	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL) .....	9-29
9.4.1.13	DDR SDRAM Data Initialization (DDR_DATA_INIT) .....	9-29
9.4.1.14	DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL) .....	9-30
9.4.1.15	DDR Initialization Address (DDR_INIT_ADDR).....	9-30
9.4.1.16	DDR Initialization Enable Extended Address (DDR_INIT_EXT_ADDR).....	9-31
9.4.1.17	DDR IP Block Revision 1 (DDR_IP_REV1).....	9-32
9.4.1.18	DDR IP Block Revision 2 (DDR_IP_REV2).....	9-32
9.4.1.19	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI).....	9-33
9.4.1.20	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO).....	9-33
9.4.1.21	Memory Data Path Error Injection Mask ECC (ERR_INJECT).....	9-34
9.4.1.22	Memory Data Path Read Capture High (CAPTURE_DATA_HI).....	9-34
9.4.1.23	Memory Data Path Read Capture Low (CAPTURE_DATA_LO) .....	9-35
9.4.1.24	Memory Data Path Read Capture ECC (CAPTURE_ECC).....	9-35
9.4.1.25	Memory Error Detect (ERR_DETECT).....	9-35
9.4.1.26	Memory Error Disable (ERR_DISABLE).....	9-36
9.4.1.27	Memory Error Interrupt Enable (ERR_INT_EN).....	9-37
9.4.1.28	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES).....	9-38
9.4.1.29	Memory Error Address Capture (CAPTURE_ADDRESS) .....	9-39
9.4.1.30	Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS).....	9-40
9.4.1.31	Single-Bit ECC Memory Error Management (ERR_SBE) .....	9-40
9.5	Functional Description.....	9-41
9.5.1	DDR SDRAM Interface Operation.....	9-45
9.5.1.1	Supported DDR SDRAM Organizations.....	9-45
9.5.2	DDR SDRAM Address Multiplexing.....	9-47
9.5.3	JEDEC Standard DDR SDRAM Interface Commands .....	9-52
9.5.4	DDR SDRAM Interface Timing.....	9-54
9.5.4.1	Clock Distribution .....	9-57
9.5.5	DDR SDRAM Mode-Set Command Timing.....	9-58
9.5.6	DDR SDRAM Registered DIMM Mode .....	9-59
9.5.7	DDR SDRAM Write Timing Adjustments .....	9-59
9.5.8	DDR SDRAM Refresh .....	9-60
9.5.8.1	DDR SDRAM Refresh Timing.....	9-61
9.5.8.2	DDR SDRAM Refresh and Power-Saving Modes .....	9-61
9.5.8.2.1	Self-Refresh in Sleep Mode.....	9-63
9.5.9	DDR Data Beat Ordering.....	9-64
9.5.10	Page Mode and Logical Bank Retention .....	9-64
9.5.11	Error Checking and Correcting (ECC) .....	9-65
9.5.12	Error Management .....	9-67
9.6	Initialization/Application Information .....	9-68

# Contents

Paragraph Number	Title	Page Number
9.6.1	Programming Differences Between Memory Types.....	9-69
9.6.2	DDR SDRAM Initialization Sequence .....	9-72
9.6.3	Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System .....	9-72
9.6.3.1	Hardware Based Self-Refresh.....	9-72
9.6.3.2	Software Based Self-Refresh.....	9-73
9.6.3.3	Bypassing Re-initialization During Battery-Backed Operation .....	9-73

## Chapter 10 Programmable Interrupt Controller

10.1	Introduction.....	10-1
10.1.1	Overview.....	10-1
10.1.2	Features.....	10-3
10.1.3	Interrupts to the Processor Core.....	10-3
10.1.4	Modes of Operation .....	10-4
10.1.4.1	Mixed Mode (GCR[M] = 1) .....	10-4
10.1.4.2	Pass-Through Mode (GCR[M] = 0) .....	10-5
10.1.5	Interrupt Sources.....	10-5
10.1.5.1	Interrupt Routing—Mixed Mode.....	10-6
10.1.5.2	Internal Interrupt Sources .....	10-6
10.2	External Signal Descriptions .....	10-7
10.2.1	Signal Overview .....	10-7
10.2.2	Detailed Signal Descriptions .....	10-8
10.3	Memory Map/Register Definition .....	10-9
10.3.1	Global Registers.....	10-17
10.3.1.1	Block Revision Register 1 (BRR1).....	10-17
10.3.1.2	Block Revision Register 2 (BRR2).....	10-18
10.3.1.3	Feature Reporting Register (FRR).....	10-18
10.3.1.4	Global Configuration Register (GCR).....	10-19
10.3.1.5	Vendor Identification Register (VIR) .....	10-20
10.3.1.6	Processor Initialization Register (PIR) .....	10-20
10.3.1.7	IPI Vector/Priority Registers (IPIVPR <sub>n</sub> ) .....	10-21
10.3.1.8	Spurious Vector Register (SVR).....	10-22
10.3.2	Global Timer Registers .....	10-22
10.3.2.1	Timer Frequency Reporting Register (TFRR).....	10-22
10.3.2.2	Global Timer Current Count Registers (GTCCR <sub>n</sub> ) .....	10-23
10.3.2.3	Global Timer Base Count Registers (GTBCR <sub>n</sub> ).....	10-23
10.3.2.4	Global Timer Vector/Priority Registers (GTVPR <sub>n</sub> ).....	10-24
10.3.2.5	Global Timer Destination Registers (GTDR <sub>n</sub> ) .....	10-25
10.3.2.6	Timer Control Register (TCR).....	10-25

# Contents

Paragraph Number	Title	Page Number
10.3.3	External, $\overline{\text{IRQ\_OUT}}$ , and Critical Interrupt Summary Registers .....	10-27
10.3.3.1	External Interrupt Summary Register (ERQSR) .....	10-27
10.3.3.2	$\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0) .....	10-28
10.3.3.3	$\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1) .....	10-29
10.3.3.4	$\overline{\text{IRQ\_OUT}}$ Summary Register 2 (IRQSR2) .....	10-29
10.3.3.5	Critical Interrupt Summary Register 0 (CISR0) .....	10-30
10.3.3.6	Critical Interrupt Summary Register 1 (CISR1) .....	10-30
10.3.3.7	Critical Interrupt Summary Register 2 (CISR2) .....	10-31
10.3.4	Performance Monitor Mask Registers (PMMRs) .....	10-31
10.3.4.1	Performance Monitor <i>n</i> Mask Registers 0 (PM <sub><i>n</i></sub> MR0) .....	10-31
10.3.4.2	Performance Monitor <i>n</i> Mask Registers 1 (PM <sub><i>n</i></sub> MR1) .....	10-32
10.3.4.3	Performance Monitor <i>n</i> Mask Registers 2 (PM <sub><i>n</i></sub> MR2) .....	10-33
10.3.5	Message Registers .....	10-33
10.3.5.1	Message Registers (MSGR0–MSGR3) .....	10-33
10.3.5.2	Message Enable Register (MER) .....	10-34
10.3.5.3	Message Status Register (MSR) .....	10-34
10.3.6	Shared Message Signaled Registers .....	10-35
10.3.6.1	Shared Message Signaled Interrupt Registers (MSIRs) .....	10-35
10.3.6.2	Shared Message Signaled Interrupt Status Register (MSISR) .....	10-35
10.3.6.3	Shared Message Signaled Interrupt Index Register (MSIIR) .....	10-36
10.3.6.4	Shared Message Signaled Interrupt Vector/Priority Register (MSIVPR <sub><i>n</i></sub> ) .....	10-37
10.3.6.5	Shared Message Signaled Interrupt Destination Register (MSIDR <sub><i>n</i></sub> ) .....	10-37
10.3.7	Interrupt Source Configuration Registers .....	10-38
10.3.7.1	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11) .....	10-38
10.3.7.2	External Interrupt Destination Registers (EIDR0–EIDR11) .....	10-39
10.3.7.3	Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR47) .....	10-40
10.3.7.4	Internal Interrupt Destination Registers (IIDR0–IIDR47) .....	10-41
10.3.7.5	Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3) .....	10-42
10.3.7.6	Messaging Interrupt Destination Registers (MIDR0–MIDR3) .....	10-42
10.3.8	Per-CPU Registers .....	10-43
10.3.8.1	Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3) .....	10-44
10.3.8.2	Processor Current Task Priority Register (CTPR) .....	10-45
10.3.8.3	Who Am I Register (WHOAMI) .....	10-46
10.3.8.4	Processor Interrupt Acknowledge Register (IACK) .....	10-46
10.3.8.5	Processor End of Interrupt Register (EOI) .....	10-47
10.4	Functional Description .....	10-47
10.4.1	Flow of Interrupt Control .....	10-47
10.4.1.1	Interrupt Source Priority .....	10-49
10.4.1.2	Processor Current Task Priority .....	10-49
10.4.1.3	Interrupt Acknowledge .....	10-49
10.4.2	Nesting of Interrupts .....	10-50

# Contents

Paragraph Number	Title	Page Number
10.4.3	Spurious Vector Generation .....	10-50
10.4.4	Messaging Interrupts.....	10-50
10.4.5	Shared Message Signaled Interrupts.....	10-50
10.4.6	PCI Express INTx.....	10-51
10.4.7	Global Timers .....	10-51
10.4.8	Reset of the PIC .....	10-52
10.5	Initialization/Application Information .....	10-52
10.5.1	Programming Guidelines .....	10-52
10.5.1.1	PIC Registers .....	10-53
10.5.1.2	Changing Interrupt Source Configuration .....	10-54

## Chapter 11 I<sup>2</sup>C Interfaces

11.1	Introduction.....	11-1
11.1.1	Overview.....	11-2
11.1.2	Features.....	11-2
11.1.3	Modes of Operation .....	11-2
11.2	External Signal Descriptions .....	11-3
11.2.1	Signal Overview .....	11-3
11.2.2	Detailed Signal Descriptions .....	11-3
11.3	Memory Map/Register Definition .....	11-4
11.3.1	Register Descriptions.....	11-5
11.3.1.1	I <sup>2</sup> C Address Register (I2CADR) .....	11-6
11.3.1.2	I <sup>2</sup> C Frequency Divider Register (I2CFDR).....	11-6
11.3.1.3	I <sup>2</sup> C Control Register (I2CCR) .....	11-7
11.3.1.4	I <sup>2</sup> C Status Register (I2CSR) .....	11-9
11.3.1.5	I <sup>2</sup> C Data Register (I2CDR).....	11-10
11.3.1.6	Digital Filter Sampling Rate Register (I2CDFSRR) .....	11-11
11.4	Functional Description.....	11-11
11.4.1	Transaction Protocol .....	11-11
11.4.1.1	START Condition .....	11-12
11.4.1.2	Slave Address Transmission.....	11-12
11.4.1.3	Repeated START Condition .....	11-13
11.4.1.4	STOP Condition.....	11-13
11.4.1.5	Protocol Implementation Details .....	11-14
11.4.1.5.1	Transaction Monitoring—Implementation Details.....	11-14
11.4.1.5.2	Control Transfer—Implementation Details .....	11-14
11.4.1.6	Address Compare—Implementation Details .....	11-15
11.4.2	Arbitration Procedure .....	11-15
11.4.2.1	Arbitration Control .....	11-15

# Contents

Paragraph Number	Title	Page Number
11.4.3	Handshaking .....	11-16
11.4.4	Clock Control.....	11-16
11.4.4.1	Clock Synchronization.....	11-16
11.4.4.2	Input Synchronization and Digital Filter .....	11-16
11.4.4.2.1	Input Signal Synchronization .....	11-17
11.4.4.2.2	Filtering of SCL and SDA Lines .....	11-17
11.4.4.3	Clock Stretching .....	11-17
11.4.5	Boot Sequencer Mode.....	11-17
11.4.5.1	EEPROM Calling Address .....	11-19
11.4.5.2	EEPROM Data Format .....	11-19
11.5	Initialization/Application Information .....	11-21
11.5.1	Initialization Sequence.....	11-21
11.5.2	Generation of START .....	11-22
11.5.3	Post-Transfer Software Response .....	11-22
11.5.4	Generation of STOP.....	11-23
11.5.5	Generation of Repeated START .....	11-23
11.5.6	Generation of SCL When SDA Low .....	11-23
11.5.7	Slave Mode Interrupt Service Routine.....	11-23
11.5.7.1	Slave Transmitter and Received Acknowledge .....	11-24
11.5.7.2	Loss of Arbitration and Forcing of Slave Mode .....	11-24
11.5.8	Interrupt Service Routine Flowchart.....	11-24

## Chapter 12 Security Engine (SEC) 2.1

12.1	SEC 2.1 Architecture Overview .....	12-2
12.1.1	Descriptors .....	12-4
12.1.2	Execution Units (EUs) .....	12-5
12.1.2.1	Public Key Execution Unit (PKEU) .....	12-5
12.1.2.1.1	Elliptic Curve Operations .....	12-5
12.1.2.1.2	Modular Exponentiation Operations .....	12-6
12.1.2.2	Data Encryption Standard Execution Unit (DEU).....	12-6
12.1.2.3	ARC Four Execution Unit (AFEU) .....	12-6
12.1.2.4	Message Digest Execution Unit (MDEU) .....	12-7
12.1.2.5	Random Number Generator (RNG).....	12-7
12.1.2.6	Advanced Encryption Standard Execution Unit (AESU).....	12-7
12.1.2.7	Kasumi Execution Unit (KEU).....	12-8
12.1.3	Crypto-Channels .....	12-8
12.1.4	Controller .....	12-9
12.1.4.1	Channel-Controlled Access .....	12-10
12.1.4.2	Host-Controlled Access .....	12-10

# Contents

Paragraph Number	Title	Page Number
12.2	Configuration of Internal Memory Space .....	12-10
12.3	Descriptor Overview .....	12-16
12.3.1	Descriptor Structure .....	12-16
12.3.2	Descriptor Format: Header Dword .....	12-17
12.3.2.1	Selecting Execution Units—EU_SEL0 and EU_SEL1 .....	12-18
12.3.2.2	Selecting Descriptor Type—DESC_TYPE .....	12-19
12.3.3	Descriptor Format: Pointer Dwords.....	12-20
12.3.4	Link Table Format .....	12-21
12.3.5	Descriptor Types .....	12-24
12.4	Execution Units.....	12-25
12.4.1	Public Key Execution Unit (PKEU) .....	12-26
12.4.1.1	PKEU Mode Register (PKEUMR).....	12-26
12.4.1.2	PKEU Key Size Register (PKEUKSR) .....	12-27
12.4.1.3	PKEU AB Size Register (PKEUABS) .....	12-28
12.4.1.4	PKEU Data Size Register (PKEUDSR) .....	12-28
12.4.1.5	PKEU Reset Control Register (PKEURCR) .....	12-29
12.4.1.6	PKEU Status Register (PKEUSR).....	12-29
12.4.1.7	PKEU Interrupt Status Register (PKEUISR).....	12-30
12.4.1.8	PKEU Interrupt Control Register (PKEUICR).....	12-31
12.4.1.9	PKEU EU Go Register (PKEUEUG) .....	12-32
12.4.1.10	PKEU Parameter Memories .....	12-32
12.4.1.10.1	PKEU Parameter Memory A .....	12-33
12.4.1.10.2	PKEU Parameter Memory B .....	12-33
12.4.1.10.3	PKEU Parameter Memory E .....	12-33
12.4.1.10.4	PKEU Parameter Memory N.....	12-33
12.4.2	Data Encryption Standard Execution Unit (DEU).....	12-33
12.4.2.1	DEU Mode Register (DEUMR) .....	12-33
12.4.2.2	DEU Key Size Register (DEUKSR).....	12-34
12.4.2.3	DEU Data Size Register (DEUDSR).....	12-35
12.4.2.4	DEU Reset Control Register (DEURCR).....	12-35
12.4.2.5	DEU Status Register (DEUSR) .....	12-36
12.4.2.6	DEU Interrupt Status Register (DEUISR) .....	12-37
12.4.2.7	DEU Interrupt Control Register (DEUICR).....	12-39
12.4.2.8	DEU EU Go Register (DEUEUG) .....	12-40
12.4.2.9	DEU IV Register (DEUIV) .....	12-41
12.4.2.10	DEU Key Registers 1–3 (DEUK <sub>n</sub> ).....	12-41
12.4.2.11	DEU FIFOs .....	12-41
12.4.3	ARC Four Execution Unit (AFEU) .....	12-41
12.4.3.1	AFEU Mode Register (AFEUMR).....	12-42
12.4.3.2	Host-Provided Context via Prevent Permute .....	12-42
12.4.3.2.1	Dump Context.....	12-42

# Contents

Paragraph Number	Title	Page Number
12.4.3.3	AFEU Key Size Register (AFEUKSR) .....	12-43
12.4.3.4	AFEU Context/Data Size Register (AFEUDSR) .....	12-44
12.4.3.5	AFEU Reset Control Register (AFEURCR) .....	12-44
12.4.3.6	AFEU Status Register (AFEUSR).....	12-45
12.4.3.7	AFEU Interrupt Status Register (AFEUISR).....	12-46
12.4.3.8	AFEU Interrupt Control Register (AFEUICR).....	12-48
12.4.3.9	AFEU EU Go Register (AFEUEUG).....	12-49
12.4.3.10	AFEU Context .....	12-49
12.4.3.10.1	AFEU Context Memory .....	12-49
12.4.3.10.2	AFEU Context Memory Pointer Register .....	12-50
12.4.3.11	AFEU Key Registers 0–1 (AFEUK <sub>n</sub> ) .....	12-50
12.4.3.12	AFEU FIFOs.....	12-50
12.4.4	Message Digest Execution Unit (MDEU) .....	12-50
12.4.4.1	MDEU Mode Register (MDEUMR) .....	12-50
12.4.4.2	Recommended Settings for MDEU Mode Register .....	12-53
12.4.4.3	MDEU Key Size Register (MDEUKSR) .....	12-54
12.4.4.4	MDEU Data Size Register (MDEUDSR).....	12-54
12.4.4.5	MDEU Reset Control Register (MDEURCR).....	12-55
12.4.4.6	MDEU Status Register (MDEUSR) .....	12-55
12.4.4.7	MDEU Interrupt Status Register (MDEUISR).....	12-57
12.4.4.8	MDEU Interrupt Control Register (MDEUICR).....	12-58
12.4.4.9	MDEU ICV Size Register (MDEUICVSR) .....	12-59
12.4.4.10	MDEU EU Go Register (MDEUEUG) .....	12-59
12.4.4.11	MDEU Context Registers.....	12-60
12.4.4.12	MDEU Key Registers .....	12-61
12.4.4.13	MDEU FIFOs .....	12-62
12.4.5	Random Number Generator (RNG).....	12-62
12.4.5.1	RNG Mode Register (RNGMR).....	12-63
12.4.5.2	RNG Data Size Register (RNGDSR) .....	12-63
12.4.5.3	RNG Reset Control Register (RNGRCR) .....	12-63
12.4.5.4	RNG Status Register (RNGSR).....	12-64
12.4.5.5	RNG Interrupt Status Register (RNGISR).....	12-65
12.4.5.6	RNG Interrupt Control Register (RNGICR).....	12-66
12.4.5.7	RNG EU Go Register (RNGEUG).....	12-67
12.4.5.8	RNG FIFO .....	12-67
12.4.6	Advanced Encryption Standard Execution Unit (AESU).....	12-67
12.4.6.1	AESU Mode Register (AESUMR).....	12-67
12.4.6.2	AESU Key Size Register (AESUKSR) .....	12-69
12.4.6.3	AESU Data Size Register (AESUDSR) .....	12-69
12.4.6.4	AESU Reset Control Register (AESURCR) .....	12-70
12.4.6.5	AESU Status Register (AESUSR).....	12-71



# Contents

Paragraph Number	Title	Page Number
12.4.6.6	AESU Interrupt Status Register (AESUISR).....	12-72
12.4.6.7	AESU Interrupt Control Register (AESUICR).....	12-73
12.4.6.8	AESU EU Go Register (AESUEUG).....	12-74
12.4.6.9	AESU Context Registers .....	12-75
12.4.6.9.1	Context for CBC Mode.....	12-76
12.4.6.9.2	Context for Counter Mode.....	12-76
12.4.6.9.3	Context for SRT Mode .....	12-76
12.4.6.9.4	Context for CCM Mode.....	12-77
12.4.6.9.5	AESU Key Registers .....	12-79
12.4.6.9.6	AESU FIFOs.....	12-79
12.4.7	Kasumi Execution Unit (KEU).....	12-79
12.4.7.1	KEU Mode Register (KEUMR) .....	12-80
12.4.7.2	KEU Key Size Register (KEUKSR).....	12-81
12.4.7.3	KEU Data Size Register (KEUDSR).....	12-81
12.4.7.4	KEU Reset Control Register (KEURCR).....	12-82
12.4.7.5	KEU Status Register (KEUSR) .....	12-83
12.4.7.6	KEU Interrupt Status Register (KEUISR) .....	12-84
12.4.7.7	KEU Interrupt Control Register (KEUICR).....	12-85
12.4.7.8	KEU Data Out Register (F9 MAC) (KEUDOR).....	12-87
12.4.7.9	KEU EU Go Register (KEUEUG) .....	12-87
12.4.7.10	KEU IV 1 Register (KEUIV1) .....	12-88
12.4.7.11	KEU ICV_In Register (KEUICV).....	12-88
12.4.7.12	KEU IV_2 Register (Fresh) (KEUIV2).....	12-89
12.4.7.13	KEU Context Data Registers (KEUC <sub>n</sub> ) .....	12-89
12.4.7.14	KEU Key Data Registers_1 and _2 (Confidentiality Key) (KEUKD <sub>n</sub> ).....	12-89
12.4.7.15	KEU Key Data Registers _3 and _4 (Integrity Key) (KEUKD <sub>n</sub> ) .....	12-90
12.4.7.16	KEU FIFOs.....	12-90
12.5	Crypto-Channels .....	12-91
12.5.1	Channel Registers .....	12-92
12.5.1.1	Crypto-Channel Configuration Registers 1–4 (CCCR <sub>n</sub> ).....	12-92
12.5.1.2	Crypto-Channel Pointer Status Registers 1–4 (CCPSR <sub>n</sub> ) .....	12-95
12.5.1.3	Crypto-Channel Current Descriptor Pointer Registers 1–4 (CCDPR <sub>n</sub> ) .....	12-100
12.5.1.4	Fetch FIFO Address Registers 1–4 (FF <sub>n</sub> ).....	12-101
12.5.1.5	Crypto-Channel 1–4 Descriptor Buffers [0–7] (DB <sub>n</sub> [0–7]) .....	12-102
12.5.1.6	Link Table Buffer Registers (Scatter or Gather)—LTB0–3 .....	12-102
12.5.2	Channel Interrupts.....	12-103
12.5.2.1	Channel Done Interrupt .....	12-103
12.5.2.2	Channel Error Interrupt.....	12-103
12.5.2.3	Channel Reset .....	12-103
12.6	Security Controller.....	12-104
12.6.1	Assignment of EUs to Channels .....	12-104

# Contents

Paragraph Number	Title	Page Number
12.6.1.1	Channel Priority Arbitration .....	12-105
12.6.1.2	Channel Round-Robin Arbitration .....	12-105
12.6.2	Bus Transfers .....	12-105
12.6.2.1	Arbitration for Use of the Controller and Buses.....	12-106
12.6.2.2	System Bus Master Reads .....	12-107
12.6.2.3	System Bus Master Writes.....	12-107
12.6.2.4	System Bus Slave Transactions (Reads and Writes) .....	12-107
12.6.3	Snooping by Caches.....	12-108
12.6.4	Controller Interrupts .....	12-108
12.6.5	Controller Registers .....	12-109
12.6.5.1	EU Assignment Status Register (EUASR) .....	12-109
12.6.5.2	Interrupt Mask Register (IMR).....	12-110
12.6.5.3	Interrupt Status Register (ISR) .....	12-111
12.6.5.4	Interrupt Clear Register (ICR) .....	12-112
12.6.5.5	ID Register.....	12-113
12.6.5.6	IP Block Revision Register.....	12-113
12.6.5.7	Master Control Register (MCR) .....	12-114
12.7	Power-Saving Mode.....	12-115

## Chapter 13 DUART

13.1	Overview.....	13-1
13.1.1	Features .....	13-1
13.1.2	Modes of Operation .....	13-2
13.2	External Signal Descriptions .....	13-3
13.2.1	Signal Overview .....	13-3
13.2.2	Detailed Signal Descriptions .....	13-3
13.3	Memory Map/Register Definition .....	13-4
13.3.1	Register Descriptions.....	13-6
13.3.1.1	Receiver Buffer Registers (URBR0, URBR1) (ULCR[DLAB] = 0) .....	13-6
13.3.1.2	Transmitter Holding Registers (UTHR0, UTHR1) (ULCR[DLAB] = 0) .....	13-6
13.3.1.3	Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (ULCR[DLAB] = 1) .....	13-7
13.3.1.4	Interrupt Enable Register (UIER) (ULCR[DLAB] = 0).....	13-9
13.3.1.5	Interrupt ID Registers (UIR0, UIR1) (ULCR[DLAB] = 0).....	13-9
13.3.1.6	FIFO Control Registers (UFCR0, UFCR1) (ULCR[DLAB] = 0).....	13-11
13.3.1.7	Line Control Registers (ULCR0, ULCR1).....	13-12
13.3.1.8	Modem Control Registers (UMCR0, UMCR1).....	13-14
13.3.1.9	Line Status Registers (ULSR0, ULSR1) .....	13-15
13.3.1.10	Modem Status Registers (UMSR0, UMSR1).....	13-16

# Contents

Paragraph Number	Title	Page Number
13.3.1.11	Scratch Registers (USCR0, USCR1) .....	13-17
13.3.1.12	Alternate Function Registers (UAFR0, UAFR1) (ULCR[DLAB] = 1) .....	13-17
13.3.1.13	DMA Status Registers (UDSR0, UDSR1) .....	13-18
13.4	Functional Description .....	13-19
13.4.1	Serial Interface .....	13-20
13.4.1.1	START Bit .....	13-20
13.4.1.2	Data Transfer .....	13-21
13.4.1.3	Parity Bit .....	13-21
13.4.1.4	STOP Bit .....	13-21
13.4.2	Baud-Rate Generator Logic .....	13-21
13.4.3	Local Loopback Mode .....	13-22
13.4.4	Errors .....	13-22
13.4.4.1	Framing Error .....	13-22
13.4.4.2	Parity Error .....	13-22
13.4.4.3	Overrun Error .....	13-22
13.4.5	FIFO Mode .....	13-22
13.4.5.1	FIFO Interrupts .....	13-23
13.4.5.2	DMA Mode Select .....	13-23
13.4.5.3	Interrupt Control Logic .....	13-23
13.5	DUART Initialization/Application Information .....	13-24

## Chapter 14 Local Bus Controller

14.1	Introduction .....	14-1
14.1.1	Overview .....	14-2
14.1.2	Features .....	14-2
14.1.3	Modes of Operation .....	14-3
14.1.3.1	LBC Bus Clock and Clock Ratios .....	14-3
14.1.3.2	Source ID Debug Mode .....	14-4
14.1.4	Power-Down Mode .....	14-4
14.2	External Signal Descriptions .....	14-4
14.3	Memory Map/Register Definition .....	14-8
14.3.1	Register Descriptions .....	14-10
14.3.1.1	Base Registers (BR0–BR7) .....	14-10
14.3.1.2	Option Registers (OR0–OR7) .....	14-12
14.3.1.2.1	Address Mask .....	14-12
14.3.1.2.2	Option Registers (OR <sub><i>n</i></sub> )—GPCM Mode .....	14-13
14.3.1.2.3	Option Registers (OR <sub><i>n</i></sub> )—UPM Mode .....	14-15
14.3.1.2.4	Option Registers (OR <sub><i>n</i></sub> )—SDRAM Mode .....	14-16
14.3.1.3	UPM Memory Address Register (MAR) .....	14-17

# Contents

Paragraph Number	Title	Page Number
14.3.1.4	UPM Mode Registers (MxMR) .....	14-17
14.3.1.5	Memory Refresh Timer Prescaler Register (MRTPR) .....	14-20
14.3.1.6	UPM Data Register (MDR) .....	14-20
14.3.1.7	SDRAM Machine Mode Register (LSDMR) .....	14-21
14.3.1.8	UPM Refresh Timer (LURT).....	14-23
14.3.1.9	SDRAM Refresh Timer (LSRT).....	14-23
14.3.1.10	Transfer Error Status Register (LTESR).....	14-24
14.3.1.11	Transfer Error Check Disable Register (LTEDR).....	14-25
14.3.1.12	Transfer Error Interrupt Enable Register (LTEIR) .....	14-26
14.3.1.13	Transfer Error Attributes Register (LTEATR) .....	14-27
14.3.1.14	Transfer Error Address Register (LTEAR).....	14-29
14.3.1.15	Local Bus Configuration Register (LBCR) .....	14-29
14.3.1.16	Clock Ratio Register (LCRR).....	14-30
14.4	Functional Description.....	14-32
14.4.1	Basic Architecture.....	14-33
14.4.1.1	Address and Address Space Checking .....	14-33
14.4.1.2	External Address Latch Enable Signal (LALE) .....	14-33
14.4.1.3	Data Transfer Acknowledge (TA) .....	14-34
14.4.1.4	Data Buffer Control (LBCTL).....	14-35
14.4.1.5	Atomic Operation .....	14-35
14.4.1.6	Parity Generation and Checking (LDP).....	14-36
14.4.1.7	Bus Monitor .....	14-36
14.4.2	General-Purpose Chip-Select Machine (GPCM).....	14-36
14.4.2.1	Timing Configuration .....	14-37
14.4.2.2	Chip-Select Assertion Timing .....	14-39
14.4.2.2.1	Programmable Wait State Configuration .....	14-40
14.4.2.2.2	Chip-Select and Write Enable Negation Timing .....	14-40
14.4.2.2.3	Relaxed Timing .....	14-41
14.4.2.2.4	Output Enable ( $\overline{LOE}$ ) Timing.....	14-43
14.4.2.2.5	Extended Hold Time on Read Accesses.....	14-43
14.4.2.3	External Access Termination ( $\overline{LGTA}$ ) .....	14-46
14.4.2.4	Boot Chip-Select Operation.....	14-46
14.4.3	SDRAM Machine .....	14-47
14.4.3.1	Supported SDRAM Configurations.....	14-47
14.4.3.2	SDRAM Power-On Initialization .....	14-48
14.4.3.3	Intel PC133 and JEDEC-Standard SDRAM Interface Commands .....	14-49
14.4.3.4	Page Hit Checking .....	14-50
14.4.3.5	Page Management.....	14-50
14.4.3.6	SDRAM Address Multiplexing .....	14-50
14.4.3.7	SDRAM Device-Specific Parameters.....	14-51
14.4.3.7.1	Precharge-to-Activate Interval.....	14-52

# Contents

Paragraph Number	Title	Page Number
14.4.3.7.2	Activate-to-Read/Write Interval .....	14-52
14.4.3.7.3	Column Address to First Data Out—CAS Latency .....	14-53
14.4.3.7.4	Last Data In to Precharge—Write Recovery .....	14-53
14.4.3.7.5	Refresh Recovery Interval (RFRC) .....	14-54
14.4.3.7.6	External Address and Command Buffers (BUFCMD).....	14-54
14.4.3.8	SDRAM Interface Timing .....	14-55
14.4.3.9	SDRAM Read/Write Transactions.....	14-57
14.4.3.10	SDRAM MODE-SET Command Timing.....	14-57
14.4.3.11	SDRAM Refresh.....	14-57
14.4.3.11.1	SDRAM Refresh Timing .....	14-58
14.4.4	User-Programmable Machines (UPMs).....	14-58
14.4.4.1	UPM Requests .....	14-59
14.4.4.1.1	Memory Access Requests.....	14-60
14.4.4.1.2	UPM Refresh Timer Requests .....	14-61
14.4.4.1.3	Software Requests—RUN Command .....	14-61
14.4.4.1.4	Exception Requests.....	14-62
14.4.4.2	Programming the UPMs .....	14-62
14.4.4.2.1	UPM Programming Example (Two Sequential Writes to the RAM Array) .....	14-63
14.4.4.2.2	UPM Programming Example (Two Sequential Reads from the RAM Array) .....	14-63
14.4.4.3	UPM Signal Timing .....	14-64
14.4.4.4	RAM Array .....	14-64
14.4.4.4.1	RAM Words.....	14-65
14.4.4.4.2	Chip-Select Signal Timing (CST <sub>n</sub> ) .....	14-67
14.4.4.4.3	Byte Select Signal Timing (BST <sub>n</sub> ) .....	14-68
14.4.4.4.4	General-Purpose Signals (GnT <sub>n</sub> , GOn).....	14-69
14.4.4.4.5	Loop Control (LOOP) .....	14-69
14.4.4.4.6	Repeat Execution of Current RAM Word (REDO).....	14-69
14.4.4.4.7	Address Multiplexing (AMX) .....	14-70
14.4.4.4.8	Data Valid and Data Sample Control (UTA) .....	14-71
14.4.4.4.9	LGPL[0:5] Signal Negation (LAST) .....	14-71
14.4.4.4.10	Wait Mechanism (WAEN).....	14-71
14.4.4.5	Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge .....	14-72
14.4.4.6	Extended Hold Time on Read Accesses .....	14-73
14.4.4.7	Memory System Interface Example Using UPM .....	14-73
14.5	Initialization/Application Information .....	14-79
14.5.1	Interfacing to Peripherals.....	14-79
14.5.1.1	Multiplexed Address/Data Bus and Non-Multiplexed Address Signals .....	14-79
14.5.1.2	Peripheral Hierarchy on the Local Bus.....	14-80
14.5.1.3	Peripheral Hierarchy on the Local Bus for Very High Bus Speeds.....	14-80

# Contents

Paragraph Number	Title	Page Number
14.5.1.4	GPCM Timings.....	14-81
14.5.2	Bus Turnaround .....	14-82
14.5.2.1	Address Phase After Previous Read .....	14-82
14.5.2.2	Read Data Phase After Address Phase .....	14-82
14.5.2.3	Read-Modify-Write Cycle for Parity Protected Memory Banks .....	14-83
14.5.2.4	UPM Cycles with Additional Address Phases.....	14-83
14.5.3	Interface to Different Port-Size Devices.....	14-83
14.5.4	Interfacing to SDRAM.....	14-85
14.5.4.1	Basic SDRAM Capabilities of the Local Bus.....	14-85
14.5.4.2	Maximum Amount of SDRAM Supported.....	14-86
14.5.4.3	SDRAM Machine Limitations.....	14-87
14.5.4.3.1	Analysis of Maximum Row Number Due to Bank Select Multiplexing.....	14-87
14.5.4.3.2	Bank Select Signals .....	14-87
14.5.4.3.3	128-Mbyte SDRAM .....	14-88
14.5.4.3.4	256-Mbyte SDRAM .....	14-90
14.5.4.3.5	512-Mbyte SDRAM .....	14-90
14.5.4.3.6	Power-Down Mode.....	14-91
14.5.4.3.7	Self-Refresh .....	14-92
14.5.4.3.8	SDRAM Timing .....	14-93
14.5.4.4	Parity Support for SDRAM .....	14-95
14.5.5	Interfacing to ZBT SRAM.....	14-96
14.5.6	Interfacing to DSP Host Ports.....	14-98
14.5.6.1	Interfacing to MSC8101 HDI16 .....	14-98
14.5.6.1.1	HDI16 Peripherals .....	14-98
14.5.6.1.2	Physical Interconnections .....	14-99
14.5.6.1.3	Supporting Burst Transfers.....	14-101
14.5.6.1.4	Host 60x Bus: HDI16 Peripheral Interface Hardware Timings.....	14-101
14.5.6.2	Interfacing to MSC8102 DSI.....	14-102
14.5.6.2.1	DSI in Asynchronous SRAM-Like Mode .....	14-102
14.5.6.2.2	DSI in Synchronous Mode .....	14-105
14.5.6.2.3	Broadcast Accesses.....	14-111

## Chapter 15 Enhanced Three-Speed Ethernet Controllers

15.1	Overview.....	15-1
15.2	Features.....	15-2
15.3	Modes of Operation .....	15-4
15.4	External Signals Description .....	15-6
15.4.1	Detailed Signal Descriptions .....	15-8
15.5	Memory Map/Register Definition .....	15-12

# Contents

Paragraph Number	Title	Page Number
15.5.1	Top-Level Module Memory Map .....	15-13
15.5.2	Detailed Memory Map.....	15-13
15.5.3	Memory-Mapped Register Descriptions.....	15-22
15.5.3.1	eTSEC General Control and Status Registers.....	15-22
15.5.3.1.1	Controller ID Register (TSEC_ID).....	15-22
15.5.3.1.2	Controller ID Register (TSEC_ID2).....	15-23
15.5.3.1.3	Interrupt Event Register (IEVENT) .....	15-24
15.5.3.1.4	Interrupt Mask Register (IMASK) .....	15-28
15.5.3.1.5	Error Disabled Register (EDIS).....	15-30
15.5.3.1.6	Ethernet Control Register (ECNTRL).....	15-31
15.5.3.1.7	Pause Time Value Register (PTV) .....	15-33
15.5.3.1.8	DMA Control Register (DMACTRL) .....	15-34
15.5.3.1.9	TBI Physical Address Register (TBIPA).....	15-35
15.5.3.2	eTSEC Transmit Control and Status Registers .....	15-36
15.5.3.2.1	Transmit Control Register (TCTRL) .....	15-36
15.5.3.2.2	Transmit Status Register (TSTAT).....	15-38
15.5.3.2.3	Default VLAN Control Word Register (DFVLAN).....	15-42
15.5.3.2.4	Transmit Interrupt Coalescing Register (TXIC).....	15-43
15.5.3.2.5	Transmit Queue Control Register (TQUEUE) .....	15-44
15.5.3.2.6	TxBD Ring 0–3 Weighting Register (TR03WT).....	15-44
15.5.3.2.7	TxBD Ring 4–7 Weighting Register (TR47WT).....	15-45
15.5.3.2.8	Transmit Data Buffer Pointer High Register (TBDBPH).....	15-46
15.5.3.2.9	Transmit Buffer Descriptor Pointers 0–7 (TBPTR0–TBPTR7) .....	15-46
15.5.3.2.10	Transmit Descriptor Base Address High Register (TBASEH).....	15-47
15.5.3.2.11	Transmit Descriptor Base Address Registers (TBASE0–TBASE7) .....	15-48
15.5.3.3	eTSEC Receive Control and Status Registers .....	15-48
15.5.3.3.1	Receive Control Register (RCTRL) .....	15-48
15.5.3.3.2	Receive Status Register (RSTAT).....	15-50
15.5.3.3.3	Receive Interrupt Coalescing Register (RXIC) .....	15-52
15.5.3.3.4	Receive Queue Control Register (RQUEUE) .....	15-53
15.5.3.3.5	Receive Bit Field Extract Control Register (RBIFX).....	15-55
15.5.3.3.6	Receive Queue Filer Table Address Register (RQFAR) .....	15-56
15.5.3.3.7	Receive Queue Filer Table Control Register (RQFCR) .....	15-57
15.5.3.3.8	Receive Queue Filer Table Property Register (RQFPR) .....	15-58
15.5.3.3.9	Maximum Receive Buffer Length Register (MRBLR) .....	15-61
15.5.3.3.10	Receive Data Buffer Pointer High Register (RBDBPH).....	15-61
15.5.3.3.11	Receive Buffer Descriptor Pointers 0–7 (RBPTR0–RBPTR7) .....	15-62
15.5.3.3.12	Receive Descriptor Base Address High Register (RBASEH).....	15-63
15.5.3.3.13	Receive Descriptor Base Address Registers (RBASE0–RBASE7) .....	15-63
15.5.3.4	MAC Functionality.....	15-64
15.5.3.4.1	Configuring the MAC .....	15-64

# Contents

Paragraph Number	Title	Page Number
15.5.3.4.2	Controlling CSMA/CD.....	15-64
15.5.3.4.3	Handling Packet Collisions .....	15-64
15.5.3.4.4	Controlling Packet Flow .....	15-65
15.5.3.4.5	Controlling PHY Links.....	15-66
15.5.3.5	MAC Registers .....	15-66
15.5.3.5.1	MAC Configuration 1 Register (MACCFG1).....	15-66
15.5.3.5.2	MAC Configuration 2 Register (MACCFG2).....	15-68
15.5.3.5.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG) .....	15-70
15.5.3.5.4	Half-Duplex Register (HAFDUP) .....	15-71
15.5.3.5.5	Maximum Frame Length Register (MAXFRM) .....	15-72
15.5.3.5.6	MII Management Configuration Register (MIIMCFG) .....	15-72
15.5.3.5.7	MII Management Command Register (MIIMCOM).....	15-73
15.5.3.5.8	MII Management Address Register (MIIMADD).....	15-74
15.5.3.5.9	MII Management Control Register (MIIMCON).....	15-74
15.5.3.5.10	MII Management Status Register (MIIMSTAT) .....	15-75
15.5.3.5.11	MII Management Indicator Register (MIIMIND).....	15-75
15.5.3.5.12	Interface Status Register (IFSTAT).....	15-76
15.5.3.5.13	MAC Station Address Part 1 Register (MACSTNADDR1) .....	15-76
15.5.3.5.14	MAC Station Address Part 2 Register (MACSTNADDR2) .....	15-77
15.5.3.5.15	MAC Exact Match Address 1–15 Part 1 Registers (MAC01ADDR1–MAC15ADDR1).....	15-78
15.5.3.5.16	MAC Exact Match Address 1–15 Part 2 Registers (MAC01ADDR2–MAC15ADDR2).....	15-78
15.5.3.6	MIB Registers.....	15-79
15.5.3.6.1	Transmit and Receive 64-Byte Frame Counter (TR64) .....	15-79
15.5.3.6.2	Transmit and Receive 65- to 127-Byte Frame Counter (TR127) .....	15-80
15.5.3.6.3	Transmit and Receive 128- to 255-Byte Frame Counter (TR255) .....	15-80
15.5.3.6.4	Transmit and Receive 256- to 511-Byte Frame Counter (TR511) .....	15-81
15.5.3.6.5	Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K) .....	15-81
15.5.3.6.6	Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX).....	15-82
15.5.3.6.7	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV) .....	15-82
15.5.3.6.8	Receive Byte Counter (RBYT).....	15-83
15.5.3.6.9	Receive Packet Counter (RPKT).....	15-83
15.5.3.6.10	Receive FCS Error Counter (RFCS) .....	15-84
15.5.3.6.11	Receive Multicast Packet Counter (RMCA) .....	15-84
15.5.3.6.12	Receive Broadcast Packet Counter (RBCA) .....	15-85
15.5.3.6.13	Receive Control Frame Packet Counter (RXCF) .....	15-85
15.5.3.6.14	Receive Pause Frame Packet Counter (RXPF).....	15-86
15.5.3.6.15	Receive Unknown Opcode Packet Counter (RXUO).....	15-86
15.5.3.6.16	Receive Alignment Error Counter (RALN) .....	15-87



# Contents

Paragraph Number	Title	Page Number
15.5.3.6.17	Receive Frame Length Error Counter (RFLR).....	15-87
15.5.3.6.18	Receive Code Error Counter (RCDE) .....	15-88
15.5.3.6.19	Receive Carrier Sense Error Counter (RCSE).....	15-88
15.5.3.6.20	Receive Undersize Packet Counter (RUND).....	15-89
15.5.3.6.21	Receive Oversize Packet Counter (ROVR).....	15-89
15.5.3.6.22	Receive Fragments Counter (RFRG) .....	15-90
15.5.3.6.23	Receive Jabber Counter (RJBR).....	15-90
15.5.3.6.24	Receive Dropped Packet Counter (RDRP).....	15-91
15.5.3.6.25	Transmit Byte Counter (TBYT) .....	15-91
15.5.3.6.26	Transmit Packet Counter (TPKT).....	15-92
15.5.3.6.27	Transmit Multicast Packet Counter (TMCA).....	15-92
15.5.3.6.28	Transmit Broadcast Packet Counter (TBCA).....	15-93
15.5.3.6.29	Transmit Pause Control Frame Counter (TXPF).....	15-93
15.5.3.6.30	Transmit Deferral Packet Counter (TDFR) .....	15-94
15.5.3.6.31	Transmit Excessive Deferral Packet Counter (TEDF) .....	15-94
15.5.3.6.32	Transmit Single Collision Packet Counter (TSCL) .....	15-95
15.5.3.6.33	Transmit Multiple Collision Packet Counter (TMCL) .....	15-95
15.5.3.6.34	Transmit Late Collision Packet Counter (TLCL).....	15-96
15.5.3.6.35	Transmit Excessive Collision Packet Counter (TXCL).....	15-96
15.5.3.6.36	Transmit Total Collision Counter (TNCL) .....	15-97
15.5.3.6.37	Transmit Drop Frame Counter (TDRP).....	15-97
15.5.3.6.38	Transmit Jabber Frame Counter (TJBR) .....	15-98
15.5.3.6.39	Transmit FCS Error Counter (TFCS) .....	15-98
15.5.3.6.40	Transmit Control Frame Counter (TXCF).....	15-99
15.5.3.6.41	Transmit Oversize Frame Counter (TOVR).....	15-99
15.5.3.6.42	Transmit Undersize Frame Counter (TUND).....	15-100
15.5.3.6.43	Transmit Fragment Counter (TFRG).....	15-100
15.5.3.6.44	Carry Register 1 (CAR1).....	15-100
15.5.3.6.45	Carry Register 2 (CAR2).....	15-103
15.5.3.6.46	Carry Mask Register 1 (CAM1).....	15-104
15.5.3.6.47	Carry Mask Register 2 (CAM2).....	15-105
15.5.3.6.48	Receive Filer Rejected Packet Counter (RREJ) .....	15-106
15.5.3.7	Hash Function Registers .....	15-107
15.5.3.7.1	Individual/Group Address Registers 0–7 (IGADDR <sub>n</sub> ) .....	15-107
15.5.3.7.2	Group Address Registers 0–7 (GADDR <sub>n</sub> ) .....	15-108
15.5.3.8	FIFO Registers.....	15-109
15.5.3.8.1	FIFO Configuration Register (FIFOCFG).....	15-109
15.5.3.9	DMA Attribute Registers.....	15-110
15.5.3.9.1	Attribute Register (ATTR).....	15-110
15.5.3.9.2	Attribute Extract Length and Extract Index Register (ATTRELI) .....	15-111
15.5.3.10	Lossless Flow Control Configuration Registers .....	15-112

# Contents

Paragraph Number	Title	Page Number
15.5.3.10.1	Receive Queue Parameters 0–7 (RQPRM0–PQPRM7).....	15-112
15.5.3.10.2	Receive Free Buffer Descriptor Pointer Registers 0–7 (RFBPTR0–RFBPTR7) .....	15-113
15.5.4	Ten-Bit Interface (TBI).....	15-114
15.5.4.1	TBI Transmit Process .....	15-114
15.5.4.1.1	Packet Encapsulation.....	15-114
15.5.4.1.2	8B10B Encoding.....	15-114
15.5.4.1.3	Preamble Shortening.....	15-114
15.5.4.2	TBI Receive Process.....	15-115
15.5.4.2.1	Synchronization .....	15-115
15.5.4.2.2	Auto-Negotiation for 1000BASE-X.....	15-115
15.5.4.3	TBI MII Set Register Descriptions .....	15-115
15.5.4.3.1	Control Register (CR).....	15-116
15.5.4.3.2	Status Register (SR).....	15-117
15.5.4.3.3	AN Advertisement Register (ANA) .....	15-118
15.5.4.3.4	AN Link Partner Base Page Ability Register (ANLPBPA).....	15-120
15.5.4.3.5	AN Expansion Register (ANEX) .....	15-121
15.5.4.3.6	AN Next Page Transmit Register (ANNPT).....	15-122
15.5.4.3.7	AN Link Partner Ability Next Page Register (ANLPANP) .....	15-123
15.5.4.3.8	Extended Status Register (EXST) .....	15-123
15.5.4.3.9	Jitter Diagnostics Register (JD).....	15-124
15.5.4.3.10	TBI Control Register (TBICON).....	15-125
15.6	Functional Description.....	15-126
15.6.1	Connecting to Physical Interfaces on Ethernet.....	15-126
15.6.1.1	Media-Independent Interface (MII).....	15-127
15.6.1.2	Reduced Media-Independent Interface (RMII) .....	15-127
15.6.1.3	Gigabit Media-Independent Interface (GMII).....	15-128
15.6.1.4	Reduced Gigabit Media-Independent Interface (RGMII) .....	15-129
15.6.1.5	Ten-Bit Interface (TBI).....	15-131
15.6.1.6	Reduced Ten-Bit Interface (RTBI) .....	15-132
15.6.1.7	Ethernet Physical Interfaces Signal Summary.....	15-133
15.6.2	Connecting to FIFO Interfaces .....	15-137
15.6.2.1	Flow Control.....	15-138
15.6.2.2	CRC Appending and Checking .....	15-138
15.6.2.3	8-Bit GMII-Style Packet FIFO Mode.....	15-139
15.6.2.4	8-Bit Encoded Packet FIFO Mode .....	15-140
15.6.2.5	FIFO Interface Signal Summary.....	15-140
15.6.3	Gigabit Ethernet Controller Channel Operation .....	15-141
15.6.3.1	Initialization Sequence.....	15-141
15.6.3.1.1	Hardware Controlled Initialization.....	15-141
15.6.3.1.2	User Initialization .....	15-141

# Contents

Paragraph Number	Title	Page Number
15.6.3.2	Soft Reset and Reconfiguring Procedure.....	15-142
15.6.3.3	Gigabit Ethernet Frame Transmission .....	15-143
15.6.3.4	Gigabit Ethernet Frame Reception .....	15-144
15.6.3.5	Ethernet Preamble Customization .....	15-145
15.6.3.5.1	User-Defined Preamble Transmission .....	15-146
15.6.3.5.2	User-Visible Preamble Reception.....	15-146
15.6.3.6	RMON Support.....	15-147
15.6.3.7	Frame Recognition.....	15-147
15.6.3.7.1	Destination Address Recognition and Frame Filtering .....	15-147
15.6.3.7.2	Hash Table Algorithm.....	15-149
15.6.3.8	Magic Packet Mode .....	15-151
15.6.3.9	Flow Control.....	15-151
15.6.3.10	Interrupt Handling .....	15-152
15.6.3.10.1	Interrupt Coalescing .....	15-153
15.6.3.10.2	Interrupt Coalescing By Frame Count Threshold.....	15-153
15.6.3.10.3	Interrupt Coalescing By Timer Threshold .....	15-154
15.6.3.11	Inter-Frame Gap Time .....	15-155
15.6.3.12	Internal and External Loop Back.....	15-155
15.6.3.13	Error-Handling Procedure.....	15-155
15.6.4	TCP/IP Off-Load .....	15-157
15.6.4.1	Frame Control Blocks.....	15-158
15.6.4.2	Transmit Path Off-Load.....	15-158
15.6.4.3	Receive Path Off-Load .....	15-160
15.6.5	Quality of Service (QoS) Provision .....	15-162
15.6.5.1	Receive Queue Filer .....	15-162
15.6.5.1.1	Filing Rules .....	15-163
15.6.5.1.2	Comparing Properties with Bit Masks.....	15-164
15.6.5.1.3	Special-Case Rules .....	15-165
15.6.5.1.4	Filer Interrupt Events.....	15-165
15.6.5.1.5	Setting Up the Receive Queue Filer Table .....	15-165
15.6.5.1.6	Filer Example—802.1p Priority Filing.....	15-166
15.6.5.1.7	Filer Example—IP Diff-Serv Code Points Filing.....	15-166
15.6.5.1.8	Filer Example—TCP and UDP Port Filing .....	15-167
15.6.5.2	Transmission Scheduling .....	15-168
15.6.5.2.1	Priority-Based Queuing (PBQ).....	15-168
15.6.5.2.2	Modified Weighted Round-Robin Queuing (MWRR) .....	15-169
15.6.6	Lossless Flow Control .....	15-170
15.6.6.1	Back Pressure Determination via Free Buffers.....	15-170
15.6.6.2	Software Use of Hardware-Initiated Back Pressure .....	15-172
15.6.6.2.1	Initialization.....	15-172
15.6.6.2.2	Operation .....	15-172

# Contents

Paragraph Number	Title	Page Number
15.6.7	Buffer Descriptors.....	15-173
15.6.7.1	Data Buffer Descriptors.....	15-173
15.6.7.2	Transmit Data Buffer Descriptors (TxBD).....	15-174
15.6.7.3	Receive Buffer Descriptors (RxBD).....	15-178
15.7	Initialization/Application Information.....	15-180
15.7.1	Interface Mode Configuration.....	15-180
15.7.1.1	MII Interface Mode.....	15-181
15.7.1.2	GMII Interface Mode.....	15-185
15.7.1.3	TBI Interface Mode.....	15-189
15.7.1.4	RGMII Interface Mode.....	15-193
15.7.1.5	RMII Interface Mode.....	15-197
15.7.1.6	RTBI Interface Mode.....	15-201
15.7.1.7	8-Bit FIFO Mode.....	15-205

## Chapter 16 DMA Controller

16.1	Introduction.....	16-1
16.1.1	Block Diagram.....	16-1
16.1.2	Overview.....	16-2
16.1.3	Features.....	16-2
16.1.4	Modes of Operation.....	16-2
16.2	External Signal Description.....	16-5
16.2.1	Signal Overview.....	16-5
16.2.2	Detailed Signal Descriptions.....	16-6
16.3	Memory Map/Register Definition.....	16-6
16.3.1	DMA Register Descriptions.....	16-9
16.3.1.1	Mode Registers (MR <sub>n</sub> ).....	16-10
16.3.1.2	Status Registers (SR <sub>n</sub> ).....	16-12
16.3.1.3	Current Link Descriptor Address Registers (CLNDAR <sub>n</sub> and ECLNDAR <sub>n</sub> ).....	16-13
16.3.1.4	Source Attributes Registers (SATR <sub>n</sub> ).....	16-15
16.3.1.5	Source Address Registers (SAR <sub>n</sub> ).....	16-16
16.3.1.6	Destination Attributes Registers (DATR <sub>n</sub> ).....	16-17
16.3.1.7	Destination Address Registers (DAR <sub>n</sub> ).....	16-18
16.3.1.8	Byte Count Registers (BCR <sub>n</sub> ).....	16-19
16.3.1.9	Next Link Descriptor Address Registers (NLNDAR <sub>n</sub> and ENLNDAR <sub>n</sub> ).....	16-19
16.3.1.10	Current List Descriptor Address Registers (CLSDAR <sub>n</sub> and ECLSDAR <sub>n</sub> ).....	16-21

# Contents

Paragraph Number	Title	Page Number
16.3.1.11	Next List Descriptor Address Registers (NLSDAR $n$ and ENLSDAR $n$ ) .....	16-22
16.3.1.12	Source Stride Registers (SSR $n$ ) .....	16-22
16.3.1.13	Destination Stride Registers (DSR $n$ ) .....	16-23
16.3.1.14	DMA General Status Register (DGSR) .....	16-24
16.4	Functional Description .....	16-25
16.4.1	DMA Channel Operation .....	16-25
16.4.1.1	Basic DMA Mode Transfer .....	16-26
16.4.1.1.1	Basic Direct Mode .....	16-26
16.4.1.1.2	Basic Direct Single-Write Start Mode .....	16-26
16.4.1.1.3	Basic Chaining Mode .....	16-27
16.4.1.1.4	Basic Chaining Single-Write Start Mode .....	16-28
16.4.1.2	Extended DMA Mode Transfer .....	16-28
16.4.1.2.1	Extended Direct Mode .....	16-28
16.4.1.2.2	Extended Direct Single-Write Start Mode .....	16-28
16.4.1.2.3	Extended Chaining Mode .....	16-29
16.4.1.2.4	Extended Chaining Single-Write Start Mode .....	16-29
16.4.1.3	External Control Mode Transfer .....	16-30
16.4.1.4	Channel Continue Mode for Cascading Transfer Chains .....	16-31
16.4.1.4.1	Basic Mode .....	16-31
16.4.1.4.2	Extended Mode .....	16-32
16.4.1.5	Channel Abort .....	16-32
16.4.1.6	Bandwidth Control .....	16-32
16.4.1.7	Channel State .....	16-32
16.4.1.8	Illustration of Stride Size and Stride Distance .....	16-33
16.4.2	DMA Transfer Interfaces .....	16-33
16.4.3	DMA Errors .....	16-33
16.4.4	DMA Descriptors .....	16-34
16.4.5	Limitations and Restrictions .....	16-37
16.5	DMA System Considerations .....	16-38
16.5.1	Unusual DMA Scenarios .....	16-40
16.5.1.1	DMA to e500 Core .....	16-40
16.5.1.2	DMA to Ethernet .....	16-40
16.5.1.3	DMA to Configuration, Control, and Status Registers .....	16-40
16.5.1.4	DMA to I <sup>2</sup> C .....	16-40
16.5.1.5	DMA to DUART .....	16-41

## Chapter 17 PCI Bus Interface

17.1	Introduction .....	17-2
------	--------------------	------

# Contents

Paragraph Number	Title	Page Number
17.1.1	Overview.....	17-2
17.1.1.1	Outbound Transactions.....	17-3
17.1.1.2	Inbound Transactions.....	17-4
17.1.2	Features.....	17-4
17.1.3	Modes of Operation.....	17-4
17.1.3.1	Host/Agent Mode Configuration.....	17-5
17.1.3.1.1	Host Mode.....	17-5
17.1.3.1.2	Agent Mode.....	17-5
17.1.3.1.3	Agent Configuration Lock Mode.....	17-5
17.1.3.2	PCI Clocking Configuration.....	17-5
17.1.3.3	PCI Arbiter (Internal/External Arbiter) Configuration.....	17-5
17.1.3.4	PCI Impedance Configuration.....	17-5
17.2	External Signal Descriptions.....	17-6
17.3	Memory Map/Register Definitions.....	17-11
17.3.1	PCI Memory-Mapped Registers.....	17-11
17.3.1.1	PCI Configuration Access Registers.....	17-14
17.3.1.1.1	PCI Configuration Address Register (CFG_ADDR).....	17-14
17.3.1.1.2	PCI Configuration Data Register (CFG_DATA).....	17-15
17.3.1.1.3	PCI Interrupt Acknowledge Register (INT_ACK).....	17-15
17.3.1.2	PCI ATMU Outbound Registers.....	17-16
17.3.1.2.1	PCI Outbound Translation Address Registers (POTAR <sub>n</sub> ).....	17-16
17.3.1.2.2	PCI Outbound Translation Extended Address Registers (POTEAR <sub>n</sub> ).....	17-16
17.3.1.2.3	PCI Outbound Window Base Address Registers (POWBAR <sub>n</sub> ).....	17-17
17.3.1.2.4	PCI Outbound Window Attributes Registers (POWAR <sub>n</sub> ).....	17-18
17.3.1.3	PCI ATMU Inbound Registers.....	17-19
17.3.1.3.1	PCI Inbound Translation Address Registers (PITAR <sub>n</sub> ).....	17-20
17.3.1.3.2	PCI Inbound Window Base Address Registers (PIWBAR <sub>n</sub> ).....	17-20
17.3.1.3.3	PCI Inbound Window Base Extended Address Registers (PIWBEAR <sub>n</sub> ).....	17-21
17.3.1.3.4	PCI Inbound Window Attributes Registers (PIWAR <sub>n</sub> ).....	17-21
17.3.1.4	PCI Error Management Registers.....	17-23
17.3.1.4.1	PCI Error Detect Register (ERR_DR).....	17-24
17.3.1.4.2	PCI Error Capture Disable Register (ERR_CAP_DR).....	17-25
17.3.1.4.3	PCI Error Enable Register (ERR_EN).....	17-26
17.3.1.4.4	PCI Error Attributes Capture Register (ERR_ATTRIB).....	17-27
17.3.1.4.5	PCI Error Address Capture Register (ERR_ADDR).....	17-28
17.3.1.4.6	PCI Error Extended Address Capture Register (ERR_EXT_ADDR).....	17-28
17.3.1.4.7	PCI Error Data Low Capture Register (ERR_DL).....	17-29
17.3.1.4.8	PCI Error Data High Capture Register (ERR_DH).....	17-29
17.3.1.4.9	PCI Gasket Timer Register (GAS_TIMR).....	17-29

# Contents

Paragraph Number	Title	Page Number
17.3.2	PCI Configuration Header .....	17-30
17.3.2.1	PCI Vendor ID Register—Offset 0x00 .....	17-30
17.3.2.2	PCI Device ID Register—Offset 0x02 .....	17-31
17.3.2.3	PCI Bus Command Register—Offset 0x04 .....	17-31
17.3.2.4	PCI Bus Status Register—Offset 0x06 .....	17-32
17.3.2.5	PCI Revision ID Register—Offset 0x08 .....	17-34
17.3.2.6	PCI Bus Programming Interface Register—Offset 0x09 .....	17-34
17.3.2.7	PCI Subclass Code Register—Offset 0x0A .....	17-35
17.3.2.8	PCI Bus Base Class Code Register—Offset 0x0B .....	17-35
17.3.2.9	PCI Bus Cache Line Size Register—Offset 0x0C .....	17-35
17.3.2.10	PCI Bus Latency Timer Register—0x0D .....	17-36
17.3.2.11	PCI Base Address Registers .....	17-36
17.3.2.12	PCI Subsystem Vendor ID Register .....	17-38
17.3.2.13	PCI Subsystem ID Register .....	17-39
17.3.2.14	PCI Bus Capabilities Pointer Register .....	17-39
17.3.2.15	PCI Bus Interrupt Line Register .....	17-39
17.3.2.16	PCI Bus Interrupt Pin Register .....	17-40
17.3.2.17	PCI Bus Minimum Grant Register (MIN_GNT) .....	17-40
17.3.2.18	PCI Bus Maximum Latency Register (MAX_LAT) .....	17-41
17.3.2.19	PCI Bus Function Register (PBFR) .....	17-41
17.3.2.20	PCI Bus Arbiter Configuration Register (PBACR) .....	17-42
17.4	Functional Description .....	17-42
17.4.1	PCI Bus Arbitration .....	17-42
17.4.1.1	PCI Bus Arbiter Operation .....	17-43
17.4.1.2	PCI Bus Parking .....	17-44
17.4.1.3	Broken Master Lock-Out .....	17-45
17.4.1.4	Power-Saving Modes and the PCI Arbiter .....	17-45
17.4.2	PCI Bus Protocol .....	17-45
17.4.2.1	Basic Transfer Control .....	17-45
17.4.2.2	PCI Bus Commands .....	17-46
17.4.2.3	Addressing .....	17-47
17.4.2.3.1	Memory Space Addressing .....	17-47
17.4.2.3.2	I/O Space Addressing .....	17-48
17.4.2.3.3	Configuration Space Addressing .....	17-48
17.4.2.4	Device Selection .....	17-48
17.4.2.5	Byte Alignment .....	17-49
17.4.2.6	Bus Driving and Turnaround .....	17-49
17.4.2.7	PCI Bus Transactions .....	17-50
17.4.2.7.1	PCI Read Transactions .....	17-50
17.4.2.7.2	PCI Write Transactions .....	17-51
17.4.2.8	Transaction Termination .....	17-52

# Contents

Paragraph Number	Title	Page Number
17.4.2.8.1	Master-Initiated Termination .....	17-52
17.4.2.8.2	Target-Initiated Termination .....	17-53
17.4.2.9	Fast Back-to-Back Transactions .....	17-56
17.4.2.10	Dual Address Cycles.....	17-56
17.4.2.11	Configuration Cycles .....	17-58
17.4.2.11.1	PCI Configuration Space Header .....	17-58
17.4.2.11.2	Host Accessing the PCI Configuration Space .....	17-60
17.4.2.11.3	Agent Accessing the PCI Configuration Space .....	17-61
17.4.2.11.4	PCI Type 0 Configuration Translation.....	17-62
17.4.2.11.5	Type 1 Configuration Translation.....	17-63
17.4.2.12	Other Bus Transactions.....	17-63
17.4.2.12.1	Interrupt-Acknowledge Transactions .....	17-63
17.4.2.12.2	Special-Cycle Transactions .....	17-64
17.4.2.13	PCI Error Functions.....	17-65
17.4.2.13.1	PCI Parity .....	17-65
17.4.2.13.2	Error Reporting.....	17-66
17.5	Initialization/Application Information .....	17-67
17.5.1	Power-On Reset Configuration Modes.....	17-67
17.5.1.1	Host Mode .....	17-68
17.5.1.2	Agent Mode .....	17-68
17.5.1.3	Agent Configuration Lock Mode.....	17-68
17.5.2	Byte Ordering .....	17-69
17.5.2.1	Byte Order for Configuration Transactions .....	17-70

## Chapter 18 PCI Express Interface Controller

18.1	Introduction.....	18-1
18.1.1	Overview.....	18-1
18.1.1.1	Outbound Transactions .....	18-2
18.1.1.2	Inbound Transactions.....	18-3
18.1.2	Features.....	18-3
18.1.3	Modes of Operation .....	18-4
18.1.3.1	Root Complex/Endpoint Modes .....	18-4
18.1.3.2	Link Width.....	18-4
18.2	External Signal Descriptions .....	18-4
18.3	Memory Map/Register Definitions.....	18-5
18.3.1	PCI Express Memory Mapped Registers.....	18-5
18.3.2	PCI Express Configuration Access Registers.....	18-9
18.3.2.1	PCI Express Configuration Address Register (PEX_CONFIG_ADDR).....	18-9
18.3.2.2	PCI Express Configuration Data Register (PEX_CONFIG_DATA).....	18-10



# Contents

Paragraph Number	Title	Page Number
18.3.2.3	PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR).....	18-11
18.3.2.4	PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR).....	18-11
18.3.2.5	PCI Express Configuration Register (PEX_CONFIG).....	18-12
18.3.3	PCI Express Power Management Event and Message Registers .....	18-13
18.3.3.1	PCI Express PME and Message Detect Register (PEX_PME_MES_DR) .....	18-13
18.3.3.2	PCI Express PME and Message Disable Register (PEX_PME_MES_DISR) .....	18-14
18.3.3.3	PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER) .....	18-16
18.3.3.4	PCI Express Power Management Command Register (PEX_PMCR).....	18-17
18.3.4	PCI Express IP Block Revision Registers .....	18-18
18.3.4.1	IP Block Revision Register 1 (PEX_IP_BLK_REV1).....	18-18
18.3.4.2	IP Block Revision Register 2 (PEX_IP_BLK_REV2).....	18-19
18.3.5	PCI Express ATMU Registers .....	18-19
18.3.5.1	PCI Express Outbound ATMU Registers .....	18-19
18.3.5.1.1	PCI Express Outbound Translation Address Registers (PEXOTAR <sub>n</sub> ) .....	18-20
18.3.5.1.2	PCI Express Outbound Translation Extended Address Registers (PEXOTEAR <sub>n</sub> ).....	18-21
18.3.5.1.3	PCI Express Outbound Window Base Address Registers (PEXOWBAR <sub>n</sub> ).....	18-21
18.3.5.1.4	PCI Express Outbound Window Attributes Registers (PEXOWAR <sub>n</sub> ).....	18-22
18.3.5.2	PCI Express Inbound ATMU Registers .....	18-24
18.3.5.2.1	EP Inbound ATMU Implementation.....	18-24
18.3.5.2.2	RC Inbound ATMU Implementation.....	18-25
18.3.5.2.3	PCI Express Inbound Translation Address Registers (PEXITAR <sub>n</sub> ).....	18-25
18.3.5.2.4	PCI Express Inbound Window Base Address Registers (PEXIWBAR <sub>n</sub> ) .....	18-26
18.3.5.2.5	PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR <sub>n</sub> ) .....	18-27
18.3.5.2.6	PCI Express Inbound Window Attributes Registers (PEXIWAR <sub>n</sub> ).....	18-27
18.3.6	PCI Express Error Management Registers .....	18-29
18.3.6.1	PCI Express Error Detect Register (PEX_ERR_DR).....	18-29
18.3.6.2	PCI Express Error Interrupt Enable Register (PEX_ERR_EN) .....	18-32
18.3.6.3	PCI Express Error Disable Register (PEX_ERR_DISR) .....	18-34
18.3.6.4	PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT) .....	18-35
18.3.6.5	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0).....	18-36
18.3.6.6	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1).....	18-38
18.3.6.7	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2).....	18-39
18.3.6.8	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3).....	18-40
18.3.7	PCI Express Configuration Space Access .....	18-42

# Contents

Paragraph Number	Title	Page Number
18.3.7.1	RC Configuration Register Access.....	18-42
18.3.7.1.1	PCI Express Configuration Access Register Mechanism.....	18-42
18.3.7.1.2	Outbound ATMU Configuration Mechanism (RC-Only) .....	18-42
18.3.7.2	EP Configuration Register Access.....	18-43
18.3.8	PCI Compatible Configuration Headers .....	18-43
18.3.8.1	Common PCI Compatible Configuration Header Registers.....	18-44
18.3.8.1.1	PCI Express Vendor ID Register—Offset 0x00 .....	18-44
18.3.8.1.2	PCI Express Device ID Register—Offset 0x02.....	18-44
18.3.8.1.3	PCI Express Command Register—Offset 0x04 .....	18-44
18.3.8.1.4	PCI Express Status Register—Offset 0x06 .....	18-46
18.3.8.1.5	PCI Express Revision ID Register—Offset 0x08.....	18-47
18.3.8.1.6	PCI Express Class Code Register—Offset 0x09 .....	18-47
18.3.8.1.7	PCI Express Cache Line Size Register—Offset 0x0C .....	18-48
18.3.8.1.8	PCI Express Latency Timer Register—0x0D.....	18-49
18.3.8.1.9	PCI Express Header Type Register—0x0E .....	18-49
18.3.8.1.10	PCI Express BIST Register—0x0F .....	18-50
18.3.8.2	Type 0 Configuration Header .....	18-50
18.3.8.2.1	PCI Express Base Address Registers—0x10–0x27.....	18-50
18.3.8.2.2	PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C .....	18-53
18.3.8.2.3	PCI Express Subsystem ID Register (EP-Mode Only)—0x2E.....	18-53
18.3.8.2.4	Capabilities Pointer Register—0x34 .....	18-54
18.3.8.2.5	PCI Express Interrupt Line Register (EP-Mode Only)—0x3C .....	18-54
18.3.8.2.6	PCI Express Interrupt Pin Register—0x3D.....	18-55
18.3.8.2.7	PCI Express Minimum Grant Register (EP-Mode Only)—0x3E .....	18-55
18.3.8.2.8	PCI Express Maximum Latency Register (EP-Mode Only)—0x3F .....	18-56
18.3.8.3	Type 1 Configuration Header .....	18-56
18.3.8.3.1	PCI Express Base Address Register 0—0x10 .....	18-57
18.3.8.3.2	PCI Express Primary Bus Number Register—Offset 0x18.....	18-57
18.3.8.3.3	PCI Express Secondary Bus Number Register—Offset 0x19.....	18-58
18.3.8.3.4	PCI Express Subordinate Bus Number Register—Offset 0x1A.....	18-58
18.3.8.3.5	PCI Express Secondary Latency Timer Register—0x1B.....	18-59
18.3.8.3.6	PCI Express I/O Base Register—0x1C .....	18-59
18.3.8.3.7	PCI Express I/O Limit Register—0x1D.....	18-59
18.3.8.3.8	PCI Express Secondary Status Register—0x1E.....	18-60
18.3.8.3.9	PCI Express Memory Base Register—0x20 .....	18-61
18.3.8.3.10	PCI Express Memory Limit Register—0x22 .....	18-61
18.3.8.3.11	PCI Express Prefetchable Memory Base Register—0x24 .....	18-62
18.3.8.3.12	PCI Express Prefetchable Memory Limit Register—0x26 .....	18-62
18.3.8.3.13	PCI Express Prefetchable Base Upper 32 Bits Register—0x28.....	18-63
18.3.8.3.14	PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C .....	18-63
18.3.8.3.15	PCI Express I/O Base Upper 16 Bits Register—0x30 .....	18-63

# Contents

Paragraph Number	Title	Page Number
18.3.8.3.16	PCI Express I/O Limit Upper 16 Bits Register—0x32 .....	18-64
18.3.8.3.17	Capabilities Pointer Register—0x34 .....	18-64
18.3.8.3.18	PCI Express Interrupt Line Register—0x3C .....	18-65
18.3.8.3.19	PCI Express Interrupt Pin Register—0x3D.....	18-65
18.3.8.3.20	PCI Express Bridge Control Register—0x3E .....	18-66
18.3.9	PCI Compatible Device-Specific Configuration Space.....	18-67
18.3.9.1	PCI Express Power Management Capability ID Register—0x44 .....	18-68
18.3.9.2	PCI Express Power Management Capabilities Register—0x46.....	18-68
18.3.9.3	PCI Express Power Management Status and Control Register—0x48 .....	18-69
18.3.9.4	PCI Express Power Management Data Register—0x4B.....	18-69
18.3.9.5	PCI Express Capability ID Register—0x4C.....	18-70
18.3.9.6	PCI Express Capabilities Register—0x4E.....	18-70
18.3.9.7	PCI Express Device Capabilities Register—0x50.....	18-71
18.3.9.8	PCI Express Device Control Register—0x54.....	18-71
18.3.9.9	PCI Express Device Status Register—0x56 .....	18-72
18.3.9.10	PCI Express Link Capabilities Register—0x58 .....	18-73
18.3.9.11	PCI Express Link Control Register—0x5C.....	18-73
18.3.9.12	PCI Express Link Status Register—0x5E .....	18-74
18.3.9.13	PCI Express Slot Capabilities Register—0x60.....	18-75
18.3.9.14	PCI Express Slot Control Register—0x64 .....	18-76
18.3.9.15	PCI Express Slot Status Register—0x66.....	18-76
18.3.9.16	PCI Express Root Control Register (RC Mode Only)—0x68.....	18-77
18.3.9.17	PCI Express Root Status Register (RC Mode Only)—0x6C.....	18-78
18.3.9.18	PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70.....	18-78
18.3.9.19	PCI Express MSI Message Control Register (EP Mode Only)—0x72 .....	18-78
18.3.9.20	PCI Express MSI Message Address Register (EP Mode Only)—0x74 .....	18-79
18.3.9.21	PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78.....	18-79
18.3.9.22	PCI Express MSI Message Data Register (EP Mode Only)—0x7C .....	18-80
18.3.10	PCI Express Extended Configuration Space .....	18-81
18.3.10.1	PCI Express Advanced Error Reporting Capability ID Register—0x100.....	18-82
18.3.10.2	PCI Express Uncorrectable Error Status Register—0x104 .....	18-82
18.3.10.3	PCI Express Uncorrectable Error Mask Register—0x108 .....	18-83
18.3.10.4	PCI Express Uncorrectable Error Severity Register—0x10C .....	18-84
18.3.10.5	PCI Express Correctable Error Status Register—0x110 .....	18-85
18.3.10.6	PCI Express Correctable Error Mask Register—0x114 .....	18-85
18.3.10.7	PCI Express Advanced Error Capabilities and Control Register—0x118 .....	18-86
18.3.10.8	PCI Express Header Log Register—0x11C–0x12B .....	18-87
18.3.10.9	PCI Express Root Error Command Register—0x12C.....	18-88
18.3.10.10	PCI Express Root Error Status Register—0x130 .....	18-88

# Contents

Paragraph Number	Title	Page Number
18.3.10.11	PCI Express Correctable Error Source ID Register—0x134.....	18-89
18.3.10.12	PCI Express Error Source ID Register—0x136 .....	18-89
18.3.10.13	LTSSM State Status Register—0x404.....	18-90
18.3.10.14	PCI Express Controller Core Clock Ratio Register—0x440.....	18-91
18.3.10.15	PCI Express Power Management Timer Register—0x450 .....	18-92
18.3.10.16	PCI Express PME Time-Out Register (EP-Mode Only)—0x454 .....	18-93
18.3.10.17	PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478.....	18-94
18.3.10.18	Configuration Ready Register—0x4B0.....	18-94
18.3.10.19	PME_To_Ack Timeout Register (RC-Mode Only)—0x590.....	18-95
18.3.10.20	Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0.....	18-95
18.4	Functional Description.....	18-97
18.4.1	Architecture .....	18-98
18.4.1.1	PCI Express Transactions .....	18-98
18.4.1.2	Byte Ordering .....	18-99
18.4.1.2.1	Byte Order for Configuration Transactions .....	18-100
18.4.1.3	Transaction Ordering Rules .....	18-101
18.4.1.4	Memory Space Addressing.....	18-101
18.4.1.5	I/O Space Addressing .....	18-101
18.4.1.6	Configuration Space Addressing .....	18-101
18.4.1.7	Serialization of Configuration and I/O Writes.....	18-102
18.4.1.8	Messages.....	18-102
18.4.1.8.1	Outbound ATMU Message Generation .....	18-102
18.4.1.8.2	Inbound Messages .....	18-104
18.4.2	Interrupts.....	18-106
18.4.2.1	EP Interrupt Generation.....	18-106
18.4.2.1.1	Hardware INTx Message Generation .....	18-106
18.4.2.1.2	Hardware MSI Generation.....	18-106
18.4.2.1.3	Software INTx Message Generation .....	18-106
18.4.2.1.4	Software MSI Generation.....	18-106
18.4.2.2	RC Handling of INTx Message and MSI Interrupt .....	18-107
18.4.2.2.1	INTx Message Handling.....	18-107
18.4.2.2.2	MSI Handling .....	18-107
18.4.3	Initial Credit Advertisement .....	18-107
18.4.4	Power Management .....	18-108
18.4.4.1	L2/L3 Ready Link State.....	18-108
18.4.5	Hot Reset.....	18-109
18.5	Initialization/Application Information .....	18-109
18.5.1	Boot Mode and Inbound Configuration Transactions .....	18-109

# Contents

Paragraph Number	Title	Page Number
<b>Part IV</b>		
<b>Global Functions and Debug</b>		
<b>Chapter 19</b>		
<b>Global Utilities</b>		
19.1	Overview.....	19-1
19.2	Global Utilities Features .....	19-1
19.2.1	Power Management and Block Disables .....	19-1
19.2.2	Accessing Current POR Configuration Settings.....	19-1
19.2.3	Interrupt and Local Bus Signal Multiplexing .....	19-1
19.2.4	Clock Control.....	19-1
19.3	External Signal Description .....	19-2
19.3.1	Signals Overview .....	19-2
19.3.2	Detailed Signal Descriptions .....	19-2
19.4	Memory Map/Register Definition .....	19-3
19.4.1	Register Descriptions.....	19-4
19.4.1.1	POR PLL Status Register (PORPLLSR) .....	19-4
19.4.1.2	POR Boot Mode Status Register (PORBMSR).....	19-6
19.4.1.3	POR I/O Impedance Status and Control Register (PORIMPSCR) .....	19-7
19.4.1.4	POR Device Status Register (PORDEVSR).....	19-8
19.4.1.5	POR Debug Mode Status Register (PORDBGMSR) .....	19-10
19.4.1.6	POR Device Status Register 2 (PORDEVSR2).....	19-10
19.4.1.7	General-Purpose POR Configuration Register (GPPORCR) .....	19-11
19.4.1.8	General-Purpose I/O Control Register (GPIOCR) .....	19-11
19.4.1.9	General-Purpose Output Data Register (GPOUTDR).....	19-12
19.4.1.10	General-Purpose Input Data Register (GPINDR).....	19-13
19.4.1.11	Alternate Function Signal Multiplex Control Register (PMUXCR) .....	19-13
19.4.1.12	Device Disable Register (DEVDISR) .....	19-14
19.4.1.13	Power Management Control and Status Register (POWMGTCSR).....	19-16
19.4.1.14	Machine Check Summary Register (MCPSUMR).....	19-17
19.4.1.15	Reset Request Status and Control Register (RSTRSCR) .....	19-18
19.4.1.16	Processor Version Register (PVR).....	19-18
19.4.1.17	System Version Register (SVR).....	19-19
19.4.1.18	Reset Control Register (RSTCR).....	19-20
19.4.1.19	LBC Voltage Select Control Register (LBCVSELCR) .....	19-20
19.4.1.20	DDR Calibration Status Register (DDRCSR) .....	19-21
19.4.1.21	DDR Control Driver Register (DDRCDR).....	19-21
19.4.1.22	DDR Clock Disable Register (DDRCLKDR) .....	19-22
19.4.1.23	Clock Out Control Register (CLKOCR) .....	19-23
19.4.1.24	SerDes 1 Control Register 1 (SRDS1CR1) .....	19-24

# Contents

Paragraph Number	Title	Page Number
19.4.1.25	SerDes 2 Control Register 1 (SRDS2CR1) .....	19-25
19.5	Functional Description.....	19-25
19.5.1	Power Management .....	19-25
19.5.1.1	Relationship Between Core and Device Power Management States.....	19-26
19.5.1.2	CKSTP_IN is Not Power Management .....	19-27
19.5.1.3	Dynamic Power Management.....	19-27
19.5.1.4	Shutting Down Unused Blocks.....	19-27
19.5.1.5	Software-Controlled Power-Down States .....	19-27
19.5.1.5.1	Doze Mode .....	19-28
19.5.1.5.2	Nap Mode .....	19-28
19.5.1.5.3	Sleep Mode .....	19-28
19.5.1.6	Power Management Control Fields .....	19-28
19.5.1.7	Power-Down Sequence Coordination.....	19-29
19.5.1.8	Interrupts and Power Management .....	19-31
19.5.1.8.1	Interrupts and Power Management Controlled by MSR[WE] .....	19-31
19.5.1.8.2	Interrupts and Power Management Controlled by POWMGTCR.....	19-31
19.5.1.9	Snooping in Power-Down Modes.....	19-32
19.5.1.10	Software Considerations for Power Management .....	19-32
19.5.1.11	Requirements for Reaching and Recovering from Sleep State.....	19-32
19.5.2	General-Purpose I/O Signals .....	19-33
19.5.3	Interrupt and Local Bus Signal Multiplexing .....	19-33

## Chapter 20 Device Performance Monitor

20.1	Introduction.....	20-1
20.1.1	Overview.....	20-2
20.1.2	Features.....	20-3
20.2	Signal Descriptions .....	20-3
20.3	Memory Map and Register Definition.....	20-3
20.3.1	Register Summary.....	20-3
20.3.2	Control Registers .....	20-5
20.3.2.1	Performance Monitor Global Control Register (PMGC0) .....	20-5
20.3.2.2	Performance Monitor Local Control Registers (PMLCAn, PMLCBn).....	20-6
20.3.3	Counter Registers.....	20-10
20.3.3.1	Performance Monitor Counters (PMC0–PMC9).....	20-10
20.4	Functional Description.....	20-11
20.4.1	Performance Monitor Interrupt.....	20-11
20.4.2	Event Counting .....	20-12
20.4.3	Threshold Events .....	20-12
20.4.4	Chaining.....	20-13

# Contents

Paragraph Number	Title	Page Number
20.4.5	Triggering .....	20-13
20.4.6	Burstiness Counting.....	20-14
20.4.7	Performance Monitor Events .....	20-16
20.4.8	Performance Monitor Examples .....	20-28

## Chapter 21 Debug Features and Watchpoint Facility

21.1	Introduction.....	21-1
21.1.1	Overview.....	21-1
21.1.2	Features.....	21-3
21.1.3	Modes of Operation .....	21-3
21.1.3.1	Local Bus (LBC) Debug Mode.....	21-4
21.1.3.2	DDR SDRAM Interface Debug Modes .....	21-4
21.1.3.3	Watchpoint Monitor Modes .....	21-4
21.1.3.4	Trace Buffer Modes .....	21-4
21.2	External Signal Description .....	21-5
21.2.1	Overview.....	21-5
21.2.2	Detailed Signal Descriptions .....	21-6
21.2.2.1	Debug Signals—Details.....	21-6
21.2.2.2	Watchpoint Monitor Trigger Signals—Details.....	21-7
21.2.2.3	Test Signals—Details.....	21-8
21.3	Memory Map/Register Definition .....	21-9
21.3.1	Watchpoint Monitor Register Descriptions .....	21-10
21.3.1.1	Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1).....	21-10
21.3.1.2	Watchpoint Monitor Address Register (WMAR).....	21-12
21.3.1.3	Watchpoint Monitor Address Mask Register (WMAMR) .....	21-14
21.3.1.4	Watchpoint Monitor Transaction Mask Register (WMTMR) .....	21-14
21.3.1.5	Watchpoint Monitor Status Register (WMSR).....	21-16
21.3.2	Trace Buffer Register Descriptions.....	21-16
21.3.2.1	Trace Buffer Control Registers (TBCR0, TBCR1) .....	21-16
21.3.2.2	Trace Buffer Address Register (TBAR) .....	21-19
21.3.2.3	Trace Buffer Address Mask Register (TBAMR).....	21-19
21.3.2.4	Trace Buffer Transaction Mask Register (TBTMR).....	21-20
21.3.2.5	Trace Buffer Status Register (TBSR) .....	21-20
21.3.2.6	Trace Buffer Access Control Register (TBACR).....	21-21
21.3.2.7	Trace Buffer Access Data High Register (TBADHR).....	21-22
21.3.2.8	Trace Buffer Access Data Register (TBADR).....	21-22
21.3.3	Context ID Registers.....	21-23
21.3.3.1	Programmed Context ID Register (PCIDR).....	21-23
21.3.3.2	Current Context ID Register (CCIDR).....	21-24

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
21.3.4	Trigger Out Function .....	21-24
21.3.4.1	Trigger Out Source Register (TOSR) .....	21-24
21.4	Functional Description.....	21-25
21.4.1	Source and Target ID .....	21-25
21.4.2	DDR SDRAM Interface Debug.....	21-26
21.4.2.1	Debug Information on Debug Pins.....	21-26
21.4.2.2	Debug Information on ECC Pins.....	21-26
21.4.3	Local Bus Interface Debug .....	21-27
21.4.4	Watchpoint Monitor .....	21-27
21.4.4.1	Watchpoint Monitor Performance Monitor Events .....	21-27
21.4.5	Trace Buffer .....	21-28
21.4.5.1	Traced Data Formats (as a Function of TBCR1[IFSEL]).....	21-28
21.5	Initialization .....	21-31

## **Appendix A Revision History**

A.1	Changes From Revision 0 to Revision 1 .....	A-1
-----	---	-----

## **Appendix B Complete List of Configuration, Control, and Status Registers**

### **Glossary**

### **Index**



# Figures

Figure Number	Title	Page Number
1-1	MPC8533E Block Diagram .....	1-2
1-2	Integrated Security Engine Functional Blocks.....	1-16
1-3	Processing Transactions Across the On-Chip Fabric.....	1-21
1-4	Multifunction Router Application Enabled by local bus, PCI Express, PCI, and Ethernet .....	1-22
1-5	Multifunction Printer Application Enabled by Local Bus, PCI Express, PCI, and Ethernet .....	1-23
1-6	Security Appliance Enabled by SEC, local bus, PCI, and Ethernet.....	1-24
1-7	IP SAN Host Adapter enabled by local bus, PCI Express, and Ethernet.....	1-25
1-8	VoIP Aggregation Application Enabled by local bus and Ethernet .....	1-26
2-1	Local Memory Map Example .....	2-2
2-2	Local Access IP Block Revision Register 1 (LAIPBRR1) .....	2-6
2-3	Local Access IP Block Revision Register 2 (LAIPBRR2) .....	2-6
2-4	Local Access Window <i>n</i> Base Address Registers (LAWBAR0–LAWBAR7) .....	2-7
2-5	Local Access Window <i>n</i> Attributes Registers (LAWAR0–LAWAR7) .....	2-7
2-6	General Utilities Registers Mapping to Configuration, Control, and Status Memory Block .....	2-12
2-7	PIC Mapping to Configuration, Control, and Status Memory Block .....	2-13
2-8	Device-Specific Register Mapping to Configuration, Control, and Status Memory Block .....	2-14
3-1	MPC8533E Signal Groupings (1/3).....	3-2
3-2	MPC8533E Signal Groupings (2/3) (Continued).....	3-3
3-3	MPC8533E Signal Groupings (3/3) (Continued).....	3-4
4-1	Configuration, Control, and Status Register Base Address Register (CCSRBAR).....	4-5
4-2	Alternate Configuration Base Address Register (ALTCBAR) .....	4-6
4-3	Alternate Configuration Attribute Register (ALTCAR) .....	4-6
4-4	Boot Page Translation Register (BPTR) .....	4-7
4-5	Power-On Reset Sequence .....	4-10
4-6	Clock Subsystem Block Diagram .....	4-22
4-7	RTC and Core Timer Facilities Clocking Options .....	4-24
5-1	e500 Core Complex Block Diagram .....	5-2
5-2	Vector and Floating-Point APUs.....	5-5
5-3	Four-Stage MU Pipeline, Showing Divide Bypass.....	5-7
5-4	Three-Stage Load/Store Unit .....	5-8
5-5	Instruction Pipeline Flow .....	5-14
5-6	GPR Issue Queue (GIQ) .....	5-15
5-7	e500 Core Programming Model.....	5-17
5-8	MMU Structure .....	5-23
5-9	Effective-to-Real Address Translation Flow.....	5-24
5-10	Effective-to-Real Address Translation Flow (e500v2) .....	5-25
6-1	Core Register Model .....	6-2

# Figures

Figure Number	Title	Page Number
6-2	Integer Exception Register (XER) .....	6-8
6-3	Condition Register (CR) .....	6-9
6-4	Link Register (LR) .....	6-11
6-5	Count Register (CTR) .....	6-11
6-6	Machine State Register (MSR) .....	6-11
6-7	Processor ID Register (PIR) .....	6-13
6-8	Processor Version Register (PVR) .....	6-13
6-9	System Version Register (SVR) .....	6-14
6-10	Timer Control Register (TCR) .....	6-14
6-11	Timer Status Register (TSR) .....	6-15
6-12	Time Base Upper/Lower Registers (TBU/TBL) .....	6-16
6-13	Decrementer Register (DEC) .....	6-16
6-14	Decrementer Auto-Reload Register (DECAR) .....	6-17
6-15	Save/Restore Register 0 (SRR0) .....	6-17
6-16	Save/Restore Register 1 (SRR1) .....	6-17
6-17	Critical Save/Restore Register 0 (CSRR0) .....	6-17
6-18	Critical Save/Restore Register 1 (CSRR1) .....	6-18
6-19	Data Exception Address Register (DEAR) .....	6-18
6-20	Interrupt Vector Prefix Register (IVPR) .....	6-18
6-21	Interrupt Vector Offset Registers (IVOR <sub>n</sub> ) .....	6-18
6-22	Exception Syndrome Register (ESR) .....	6-19
6-23	Machine Check Save/Restore Register 0 (MCSRR0) .....	6-20
6-24	Machine Check Save/Restore Register 1 (MCSRR1) .....	6-20
6-25	Machine Check Address Register (MCAR) .....	6-21
6-26	Machine Check Address Register Upper (MCARU) .....	6-21
6-27	Machine Check Syndrome Register (MCSR) .....	6-21
6-28	Software-Use SPRs (SPRG0–SPRG7 and USPRG0) .....	6-22
6-29	Branch Buffer Entry Address Register (BBEAR) .....	6-23
6-30	Branch Buffer Target Address Register (BBTAR) .....	6-23
6-31	Branch Unit Control and Status Register (BUCSR) .....	6-24
6-32	Hardware Implementation-Dependent Register 0 (HID0) .....	6-25
6-33	Hardware Implementation-Dependent Register 1 (HID1) .....	6-26
6-34	L1 Cache Control and Status Register 0 (L1CSR0) .....	6-28
6-35	L1 Cache Control and Status Register 1 (L1CSR1) .....	6-29
6-36	L1 Cache Configuration Register 0 (L1CFG0) .....	6-30
6-37	L1 Cache Configuration Register 1 (L1CFG1) .....	6-31
6-38	Process ID Registers (PID0–PID2) .....	6-32
6-39	MMU Control and Status Register 0 (MMUCSR0) .....	6-32
6-40	MMU Configuration Register (MMUCFG) .....	6-32
6-41	TLB Configuration Register 0 (TLB0CFG) .....	6-33
6-42	TLB Configuration Register 1 (TLB1CFG) .....	6-34

# Figures

Figure Number	Title	Page Number
6-43	MAS Register 0 (MAS0) .....	6-35
6-44	MAS Register 1 (MAS1) .....	6-35
6-45	MAS Register 2 (MAS2) .....	6-36
6-46	MAS Register 3 (MAS3) .....	6-37
6-47	MAS Register 4 (MAS4) .....	6-38
6-48	MAS Register 6 (MAS6) .....	6-38
6-49	MAS Register 7 (MAS7) .....	6-39
6-50	Debug Control Register 0 (DBCR0) .....	6-39
6-51	Debug Control Register 1 (DBCR1) .....	6-41
6-52	Debug Control Register 2 (DBCR2) .....	6-42
6-53	Debug Status Register (DBSR) .....	6-43
6-54	Instruction Address Compare Registers (IAC1–IAC2) .....	6-45
6-55	Data Address Compare Registers (DAC1–DAC2) .....	6-45
6-56	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR) .....	6-45
6-57	Accumulator (ACC) .....	6-47
6-58	Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0) .....	6-49
6-59	Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3) .....	6-50
6-60	Local Control B Registers (PMLCb0–PMLCb3)/User Local Control B Registers (UPMLCb0–UPMLCb3) .....	6-51
6-61	Performance Monitor Counter Registers (PMC0–PMC3)/User Performance Monitor Counter Registers (UPMC0–UPMC3) .....	6-52
7-1	L2 Cache/SRAM Configuration .....	7-1
7-2	Cache Organization .....	7-4
7-3	Physical Address Usage for L2 Cache Accesses .....	7-5
7-4	Physical Address Usage for SRAM Accesses .....	7-6
7-5	Data Bus Connection of CCB .....	7-8
7-6	Address Bus Connection of CCB .....	7-8
7-7	L2 Control Register (L2CTL) .....	7-10
7-8	Cache External Write Address Registers (L2CEWAR <sub>n</sub> ) .....	7-13
7-9	Cache External Write Address Registers Extended Address (L2CEWAREA <sub>n</sub> ) .....	7-14
7-10	Cache External Write Control Registers (L2CEWCR0–L2CEWCR3) .....	7-14
7-11	L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR <sub>n</sub> ) .....	7-16
7-12	L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA <sub>n</sub> ) .....	7-17
7-13	L2 Error Injection Mask High Register (L2ERRINJHI) .....	7-18
7-14	L2 Error Injection Mask Low Register (L2ERRINJLO) .....	7-18
7-15	L2 Error Injection Mask Control Register (L2ERRINJCTL) .....	7-19
7-16	L2 Error Capture Data High Register (L2CAPTDATAHI) .....	7-20

# Figures

Figure Number	Title	Page Number
7-17	L2 Error Capture Data Low Register (L2CAPTDATALO) .....	7-20
7-18	L2 Error Syndrome Register (L2CAPTECC) .....	7-20
7-19	L2 Error Detect Register (L2ERRDET) .....	7-21
7-20	L2 Error Disable Register (L2ERRDIS) .....	7-22
7-21	L2 Error Interrupt Enable Register (L2ERRINTEN) .....	7-22
7-22	L2 Error Attributes Capture Register (L2ERRATTR) .....	7-23
7-23	L2 Error Address Capture Register (L2ERRADDRH) .....	7-24
7-24	L2 Error Address Capture Register (L2ERRADDRL) .....	7-25
7-25	L2 Error Control Register (L2ERRCTL) .....	7-25
7-26	L2 Cache Line Replacement Algorithm .....	7-31
8-1	e500 Coherency Module Block Diagram .....	8-1
8-2	ECM CCB Address Configuration Register (EEBACR) .....	8-3
8-3	ECM CCB Port Configuration Register (EEBPCR) .....	8-4
8-4	ECM IP Block Revision Register 1 (EIPBRR1) .....	8-5
8-5	ECM IP Block Revision Register 2 (EIPBRR2) .....	8-5
8-6	ECM Error Detect Register (EEDR) .....	8-6
8-7	ECM Error Enable Register (EEER) .....	8-7
8-8	ECM Error Attributes Capture Register (EEATR) .....	8-7
8-9	ECM Error Low Address Capture Register (EELADR) .....	8-8
8-10	ECM Error High Address Capture Register (EEHADR) .....	8-9
9-1	DDR Memory Controller Simplified Block Diagram .....	9-2
9-2	Chip Select Bounds Registers (CS <sub>n</sub> _BNDS) .....	9-11
9-3	Chip Select Configuration Register (CS <sub>n</sub> _CONFIG) .....	9-12
9-4	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3) .....	9-13
9-5	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0) .....	9-14
9-6	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1) .....	9-16
9-7	DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2) .....	9-18
9-8	DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG) .....	9-20
9-9	DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2) .....	9-23
9-10	DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE) .....	9-25
9-11	DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2) .....	9-26
9-12	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL) .....	9-26
9-13	DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL) .....	9-29
9-14	DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT) .....	9-29
9-15	DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL) .....	9-30
9-16	DDR Initialization Address Configuration Register (DDR_INIT_ADDR) .....	9-31
9-17	DDR Initialization Extended Address Configuration Register (DDR_INIT_EXT_ADDR) .....	9-31
9-18	DDR IP Block Revision 1 (DDR_IP_REV1) .....	9-32
9-19	DDR IP Block Revision 2 (DDR_IP_REV2) .....	9-32

# Figures

Figure Number	Title	Page Number
9-20	Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI) .....	9-33
9-21	Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO).....	9-33
9-22	Memory Data Path Error Injection Mask ECC Register (ERR_INJECT).....	9-34
9-23	Memory Data Path Read Capture High Register (CAPTURE_DATA_HI).....	9-34
9-24	Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO) .....	9-35
9-25	Memory Data Path Read Capture ECC Register (CAPTURE_ECC).....	9-35
9-26	Memory Error Detect Register (ERR_DETECT).....	9-36
9-27	Memory Error Disable Register (ERR_DISABLE).....	9-36
9-28	Memory Error Interrupt Enable Register (ERR_INT_EN).....	9-37
9-29	Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES).....	9-38
9-30	Memory Error Address Capture Register (CAPTURE_ADDRESS) .....	9-39
9-31	Memory Error Extended Address Capture Register (CAPTURE_EXT_ADDRESS).....	9-40
9-32	Single-Bit ECC Memory Error Management Register (ERR_SBE) .....	9-40
9-33	DDR Memory Controller Block Diagram .....	9-42
9-34	Typical Dual Data Rate SDRAM Internal Organization.....	9-43
9-35	Typical DDR SDRAM Interface Signals .....	9-43
9-36	Example 256-Mbyte DDR SDRAM Configuration With ECC .....	9-44
9-37	DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2 .....	9-56
9-38	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTOR .....	9-56
9-39	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3.....	9-56
9-40	DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4 .....	9-57
9-41	DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs .....	9-58
9-42	DDR SDRAM Mode-Set Command Timing .....	9-58
9-43	Registered DDR SDRAM DIMM Burst Write Timing .....	9-59
9-44	Write Timing Adjustments Example for Write Latency = 1 .....	9-60
9-45	DDR SDRAM Bank Staggered Auto Refresh Timing.....	9-61
9-46	DDR SDRAM Power-Down Mode .....	9-62
9-47	DDR SDRAM Self-Refresh Entry Timing .....	9-63
9-48	DDR SDRAM Self-Refresh Exit Timing .....	9-63
10-1	Interrupt Sources Block Diagram .....	10-2
10-2	Pass-Through Mode Example .....	10-5
10-3	Block Revision Register 1 (BRR1).....	10-17
10-4	Block Revision Register 2 (BRR2).....	10-18
10-5	Feature Reporting Register (FRR) .....	10-18
10-6	Global Configuration Register (GCR) .....	10-19
10-7	Vendor Identification Register (VIR).....	10-20
10-8	Processor Initialization Register (PIR) .....	10-20
10-9	IPI Vector/Priority Register (IPIVPR <sub>n</sub> ) .....	10-21
10-10	Spurious Vector Register (SVR) .....	10-22
10-11	Timer Frequency Reporting Register (TFRR) .....	10-22
10-12	Global Timer Current Count Registers (GTCCR <sub>n</sub> ).....	10-23

# Figures

Figure Number	Title	Page Number
10-13	Global Timer Base Count Register (GTBCR $n$ ).....	10-23
10-14	Global Timer Vector/Priority Register (GTVPR $n$ ).....	10-24
10-15	Global Timer Destination Registers (GTDR $n$ ).....	10-25
10-16	Example Calculation for Cascaded Timers.....	10-26
10-17	Timer Control Register (TCR).....	10-26
10-18	External Interrupt Summary Register (ERQSR).....	10-28
10-19	<u>IRQ_OUT</u> Summary Register 0 (IRQSR0).....	10-28
10-20	<u>IRQ_OUT</u> Summary Register 1 (IRQSR1).....	10-29
10-21	<u>IRQ_OUT</u> Summary Register 2 (IRQSR2).....	10-29
10-22	Critical Interrupt Summary Register 0 (CISR0).....	10-30
10-23	Critical Interrupt Summary Register 1 (CISR1).....	10-30
10-24	Critical Interrupt Summary Register 2 (CISR2).....	10-31
10-25	Performance Monitor $n$ Mask Registers 0 (PM $n$ MR0).....	10-31
10-26	Performance Monitor $n$ Mask Registers 1 (PM $n$ MR1).....	10-32
10-27	Performance Monitor $n$ Mask Registers 2 (PM $n$ MR2).....	10-33
10-28	Message Registers (MSGRs).....	10-33
10-29	Message Enable Register (MER).....	10-34
10-30	Message Status Register (MSR).....	10-34
10-31	Shared Message Signaled Interrupt Register (MSIRs).....	10-35
10-32	Shared Message Signaled Interrupt Status Register (MSISR).....	10-36
10-33	Shared Message Signaled Interrupt Index Register (MSIIR).....	10-36
10-34	Shared Message Signaled Interrupt Vector/Priority Register (MSIVPR $n$ ).....	10-37
10-35	Shared Message Signaled Interrupt Destination Registers (MSIDR $n$ ).....	10-37
10-36	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11).....	10-38
10-37	External Interrupt Destination Registers (EIDRs).....	10-39
10-38	Internal Interrupt Vector/Priority Registers (IIVPRs).....	10-40
10-39	Internal Interrupt Destination Registers (IIDRs).....	10-41
10-40	Messaging Interrupt Vector/Priority Registers (MIVPRs).....	10-42
10-41	Messaging Interrupt Destination Registers (MIDRs).....	10-43
10-42	Per-CPU Register Address Decoding in a Four-Core Device.....	10-44
10-43	Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3).....	10-45
10-44	Processor Current Task Priority Register (CTPR).....	10-45
10-45	Processor Who Am I Register (WHOAMI).....	10-46
10-46	Processor Interrupt Acknowledge Register (IACK).....	10-46
10-47	End of Interrupt Register (EOI).....	10-47
10-48	PIC Interrupt Processing Flow Diagram.....	10-48
11-1	I <sup>2</sup> C Block Diagram.....	11-1
11-2	I <sup>2</sup> C Address Register (I2CADR).....	11-6
11-3	I <sup>2</sup> C Frequency Divider Register (I2CFDR).....	11-6
11-4	I <sup>2</sup> C Control Register (I2CCR).....	11-7
11-5	I <sup>2</sup> C Status Register (I2CSR).....	11-9

# Figures

Figure Number	Title	Page Number
11-6	I <sup>2</sup> C Data Register (I2CDR).....	11-10
11-7	I <sup>2</sup> C Digital Filter Sampling Rate Register (I2CDFSRR).....	11-11
11-8	I <sup>2</sup> C Interface Transaction Protocol.....	11-12
11-9	EEPROM Data Format for One Register Preload Command.....	11-20
11-10	EEPROM Contents .....	11-20
11-11	Example I <sup>2</sup> C Interrupt Service Routine Flowchart.....	11-25
12-1	SEC Connected to System Bus .....	12-3
12-2	SEC Functional Modules .....	12-3
12-3	Descriptor Format .....	12-16
12-4	Header Dword .....	12-17
12-5	Pointer Dword .....	12-20
12-6	Link Table Entry .....	12-21
12-7	Descriptors, Link Tables, and Data Parcels .....	12-23
12-8	PKEU Mode Register.....	12-26
12-9	PKEU AB Size Register .....	12-28
12-10	PKEU Data Size Register .....	12-28
12-11	PKEU Reset Control Register.....	12-29
12-12	PKEU Status Register .....	12-29
12-13	PKEU Interrupt Status Register .....	12-30
12-14	DEU Key Size Register.....	12-34
12-15	DEU Data Size Register.....	12-35
12-16	DEU Reset Control Register .....	12-35
12-17	DEU Status Register .....	12-36
12-18	DEU Interrupt Status Register .....	12-37
12-19	DEU Interrupt Control Register.....	12-39
12-20	DEU EU Go Register .....	12-41
12-21	AFEU Mode Register.....	12-42
12-22	AFEU Key Size Register .....	12-43
12-23	AFEU Context/Data Size Register.....	12-44
12-24	AFEU Reset Control Register.....	12-45
12-25	AFEU Status Register .....	12-45
12-26	AFEU Interrupt Status Register .....	12-46
12-27	AFEU Interrupt Control Register.....	12-48
12-28	AFEU EU Go Register.....	12-49
12-29	MDEU Mode Register in Old Configuration (NEW = 0).....	12-51
12-30	MDEU Mode Register in New Configuration (NEW = 1) .....	12-52
12-31	MDEU Key Size Register .....	12-54
12-32	MDEU Data Size Register .....	12-55
12-33	MDEU Reset Control Register .....	12-55
12-34	MDEU Status Register .....	12-56
12-35	MDEU Interrupt Status Register .....	12-57

# Figures

Figure Number	Title	Page Number
12-36	MDEU Interrupt Control Register .....	12-58
12-37	MDEU ICV Size Register .....	12-59
12-38	MDEU EU Go Register .....	12-60
12-39	MDEU Context Register .....	12-61
12-40	RNG Mode Register .....	12-63
12-41	RNG Data Size Register .....	12-63
12-42	RNG Reset Control Register .....	12-63
12-43	RNG Status Register .....	12-64
12-44	RNG Interrupt Status Register .....	12-65
12-45	RNG Interrupt Control Register .....	12-66
12-46	RNG EU Go Register .....	12-67
12-47	AESU Mode Register .....	12-67
12-48	AESU Key Size Register .....	12-69
12-49	AESU Data Size Register .....	12-70
12-50	AESU Reset Control Register .....	12-70
12-51	AESU Status Register .....	12-71
12-52	AESU Interrupt Status Register .....	12-72
12-53	AESU Interrupt Control Register .....	12-73
12-54	AESU EU Go Register .....	12-75
12-55	AESU Context Register .....	12-75
12-56	AESU CCM Context Registers .....	12-77
12-57	KEU Mode Register .....	12-80
12-58	KEU Key Size Register .....	12-81
12-59	KEU Data Size Register .....	12-82
12-60	KEU Reset Control Register .....	12-82
12-61	KEU Status Register .....	12-83
12-62	KEU Interrupt Status Register .....	12-84
12-63	KEU Interrupt Control Register .....	12-86
12-64	KEU Data Out Register (F9 MAC) .....	12-87
12-65	KEU EU Go Register .....	12-88
12-66	KEU IV_1 Register .....	12-88
12-67	KEU IV_2 Register (Fresh) .....	12-89
12-68	KEU Key Data Register_1 (CK-high) .....	12-90
12-69	KEU Key Data Register_2 (CK-Low) .....	12-90
12-70	KEU Key Data Register_3 (IK-high) .....	12-90
12-71	KEU Key Data Register_4 (IK-low) .....	12-90
12-72	Crypto-Channel Configuration Register (CCCR) .....	12-92
12-73	Header Dword Writeback Format .....	12-94
12-74	Crypto-Channel Pointer Status Register .....	12-95
12-75	Crypto-Channel Current Descriptor Pointer Register .....	12-100
12-76	Fetch FIFO .....	12-101



# Figures

Figure Number	Title	Page Number
12-77	Descriptor Buffer Format.....	12-102
12-78	Link Table Buffer.....	12-103
12-79	EU Assignment Status Register (EUASR).....	12-109
12-80	Interrupt Mask Register (IMR).....	12-110
12-81	Interrupt Status Register (ISR).....	12-111
12-82	Interrupt Clear Register (ICR).....	12-112
12-83	ID Register (ID).....	12-113
12-84	IP Block Revision Register.....	12-113
12-85	Master Control Register (MCR).....	12-114
13-1	UART Block Diagram.....	13-2
13-2	Receiver Buffer Registers (URBR0, URBR1).....	13-6
13-3	Transmitter Holding Registers (UTHR0, UTHR1).....	13-7
13-4	Divisor Most Significant Byte Registers (UDMB0, UDMB1).....	13-7
13-5	Divisor Least Significant Byte Registers (UDLB0, UDLB1).....	13-8
13-6	Interrupt Enable Register (UIER).....	13-9
13-7	Interrupt ID Registers (UIIR).....	13-10
13-8	FIFO Control Registers (UFCR0, UFCR1).....	13-11
13-9	Line Control Register (ULCR).....	13-13
13-10	Modem Control Register (UMCR).....	13-14
13-11	Line Status Register (ULSR).....	13-15
13-12	Modem Status Register (UMSR).....	13-16
13-13	Scratch Register (USCR).....	13-17
13-14	Alternate Function Register (UAFR).....	13-17
13-15	DMA Status Register (UDSR).....	13-18
13-16	UART Bus Interface Transaction Protocol Example.....	13-20
14-1	Local Bus Controller Block Diagram.....	14-1
14-2	Base Registers (BR <sub>n</sub> ).....	14-10
14-3	Option Registers (OR <sub>n</sub> ) in GPCM Mode.....	14-13
14-4	Option Registers (OR <sub>n</sub> ) in UPM Mode.....	14-15
14-5	Option Registers (OR <sub>n</sub> ) in SDRAM Mode.....	14-16
14-6	UPM Memory Address Register (MAR).....	14-17
14-7	UPM Mode Registers (M <sub>x</sub> MR).....	14-17
14-8	Memory Refresh Timer Prescaler Register (MRTPR).....	14-20
14-9	UPM Data Register (MDR).....	14-20
14-10	SDRAM Machine Mode Register (LSDMR).....	14-21
14-11	UPM Refresh Timer (LURT).....	14-23
14-12	LSRT SDRAM Refresh Timer (LSRT).....	14-23
14-13	Transfer Error Status Register (LTESR).....	14-24
14-14	Transfer Error Check Disable Register (LTEDR).....	14-25
14-15	Transfer Error Interrupt Enable Register (LTEIR).....	14-26
14-16	Transfer Error Attributes Register (LTEATR).....	14-27

# Figures

Figure Number	Title	Page Number
14-17	Transfer Error Address Register (LTEAR) .....	14-29
14-18	Local Bus Configuration Register.....	14-29
14-19	Clock Ratio Register (LCRR).....	14-30
14-20	Basic Operation of Memory Controllers in the LBC.....	14-32
14-21	Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420 .....	14-34
14-22	Basic LBC Bus Cycle with LALE, TA, and $\overline{LCS}_n$ .....	14-35
14-23	Local Bus to GPCM Device Interface .....	14-37
14-24	GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0) .....	14-37
14-25	GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0).....	14-40
14-26	GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, EHTR = 0, TRLX = 1) .....	14-41
14-27	GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1) .....	14-42
14-28	GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1) .....	14-42
14-29	GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1) .....	14-43
14-30	GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing).....	14-44
14-31	GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads).....	14-44
14-32	GPCM Read Followed by Write (TRLX = 1, EHTR = 0, 4-Cycle Extended Hold Time on Reads) .....	14-45
14-33	External Termination of GPCM Access.....	14-46
14-34	Connection to a 32-Bit SDRAM with 12 Address Lines.....	14-48
14-35	SDRAM Address Multiplexing .....	14-51
14-36	PRETOACT = 2 (2 Clock Cycles).....	14-52
14-37	ACTTORW = 2 (2 Clock Cycles).....	14-52
14-38	CL = 2 (2 Clock Cycles) .....	14-53
14-39	WRC = 2 (2 Clock Cycles) .....	14-53
14-40	RFRC = 4 (6 Clock Cycles) .....	14-54
14-41	BUFCMD = 1, LCRR[BUFCMDC] = 2.....	14-54
14-42	SDRAM Single-Beat Read, Page Closed, CL = 3 .....	14-55
14-43	SDRAM Single-Beat Read, Page Hit, CL = 3 .....	14-55
14-44	SDRAM Two-Beat Burst Read, Page Closed, CL = 3.....	14-55
14-45	SDRAM Four-Beat Burst Read, Page Miss, CL = 3.....	14-55
14-46	SDRAM Single-Beat Write, Page Hit.....	14-56
14-47	SDRAM Three-Beat Write, Page Closed.....	14-56
14-48	SDRAM Read-After-Read Pipelined, Page Hit, CL = 3.....	14-56
14-49	SDRAM Write-After-Write Pipelined, Page Hit.....	14-56
14-50	SDRAM Read-After-Write Pipelined, Page Hit .....	14-57

# Figures

Figure Number	Title	Page Number
14-51	SDRAM MODE-SET Command.....	14-57
14-52	SDRAM Bank-Staggered Auto-Refresh Timing .....	14-58
14-53	User-Programmable Machine Functional Block Diagram.....	14-59
14-54	RAM Array Indexing .....	14-60
14-55	Memory Refresh Timer Request Block Diagram .....	14-61
14-56	UPM Clock Scheme .....	14-64
14-57	RAM Array and Signal Generation .....	14-64
14-58	RAM Word Field Descriptions .....	14-65
14-59	$\overline{\text{LCSn}}$ Signal Selection .....	14-68
14-60	$\overline{\text{LBS}}$ Signal Selection .....	14-68
14-61	UPM Read Access Data Sampling.....	14-71
14-62	Effect of LUPWAIT Signal.....	14-72
14-63	Single-Beat Read Access to FPM DRAM .....	14-74
14-64	Single-Beat Write Access to FPM DRAM .....	14-75
14-65	Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown).....	14-76
14-66	Refresh Cycle (CBR) to FPM DRAM .....	14-77
14-67	Exception Cycle .....	14-78
14-68	Multiplexed Address/Data Bus .....	14-79
14-69	Local Bus Peripheral Hierarchy .....	14-80
14-70	Local Bus Peripheral Hierarchy for Very High Bus Speeds .....	14-81
14-71	GPCM Address Timings .....	14-81
14-72	GPCM Data Timings.....	14-82
14-73	Interface to Different Port-Size Devices .....	14-84
14-74	128-Mbyte SDRAM Diagram.....	14-88
14-75	SDRAM Power-Down Timing.....	14-92
14-76	SDRAM Self-Refresh Mode Timing .....	14-93
14-77	Local Bus PLL Operation .....	14-95
14-78	Parity Support for SDRAM.....	14-96
14-79	Interface to ZBT SRAM .....	14-97
14-80	MSC8101 HDI16 Peripheral Registers.....	14-99
14-81	Interface to MSC8101 HDI16.....	14-100
14-82	Interface to MSC8102 DSI in Asynchronous Mode .....	14-103
14-83	Asynchronous Write to MSC8102 DSI.....	14-104
14-84	Asynchronous Read from MSC8102 DSI.....	14-105
14-85	Interface to MSC8102 DSI in Synchronous Mode .....	14-106
14-86	UPM Synchronization Cycle .....	14-107
14-87	Synchronous Single Write to MSC8102 DSI.....	14-108
14-88	Synchronous Single Read from MSC8102 DSI.....	14-109
14-89	Synchronous Burst Write to MSC8102 DSI .....	14-110
14-90	Synchronous Burst Read from MSC8102 DSI .....	14-111
15-1	eTSEC Block Diagram.....	15-2

# Figures

Figure Number	Title	Page Number
15-2	TSEC_ID Register .....	15-23
15-3	TSEC_ID2 Register .....	15-23
15-4	IEVENT Register Definition .....	15-25
15-5	IMASK Register Definition .....	15-28
15-6	EDIS Register Definition .....	15-30
15-7	ECNTRL Register Definition .....	15-31
15-8	PTV Register Definition.....	15-33
15-9	DMACTRL Register.....	15-34
15-10	TBIPA Register Definition.....	15-36
15-11	TCTRL Register Definition .....	15-36
15-12	TSTAT Register Definition .....	15-38
15-13	DFVLAN Register Definition.....	15-42
15-14	TXIC Register Definition.....	15-43
15-15	TQUEUE Register Definition.....	15-44
15-16	TR03WT Register Definition.....	15-45
15-17	TR47WT Register Definition.....	15-45
15-18	TBDBPH Register Definition .....	15-46
15-19	TBPTR0–TBPTR7 Register Definition .....	15-47
15-20	TBASEH Register Definition .....	15-47
15-21	TBASE Register Definition .....	15-48
15-22	RCTRL Register Definition.....	15-48
15-23	RSTAT Register Definition .....	15-51
15-24	RXIC Register Definition .....	15-52
15-25	RQUEUE Register Definition.....	15-53
15-26	RBIFX Register Definition .....	15-55
15-27	Receive Queue Filer Table Address Register Definition .....	15-56
15-28	Receive Queue Filer Table Control Register Definition .....	15-57
15-29	Receive Queue Filer Table Property IDs 0, 2–15 Register Definition.....	15-58
15-30	Receive Queue Filer Table Property ID1 Register Definition .....	15-58
15-31	MRBLR Register Definition.....	15-61
15-32	RBDBPH Register Definition.....	15-62
15-33	RBPTR0–RBPTR7 Register Definition .....	15-62
15-34	RBASEH Register Definition .....	15-63
15-35	RBASE Register Definition .....	15-63
15-36	MACCFG1 Register Definition .....	15-66
15-37	MACCFG2 Register Definition.....	15-68
15-38	IPGIFG Register Definition .....	15-70
15-39	Half-Duplex Register Definition.....	15-71
15-40	Maximum Frame Length Register Definition.....	15-72
15-41	MII Management Configuration Register Definition .....	15-72
15-42	MIIMCOM Register Definition .....	15-73

# Figures

Figure Number	Title	Page Number
15-43	MIIMADD Register Definition .....	15-74
15-44	MII Mgmt Control Register Definition.....	15-74
15-45	MIIMSTAT Register Definition.....	15-75
15-46	MII Mgmt Indicator Register Definition .....	15-75
15-47	Interface Status Register Definition.....	15-76
15-48	MAC Station Address Part 1 Register Definition.....	15-77
15-49	MAC Station Address Part 2 Register Definition.....	15-77
15-50	MAC Exact Match Address <i>n</i> Part 1 Register Definition.....	15-78
15-51	MAC Exact Match Address <i>x</i> Part 2 Register Definition.....	15-78
15-52	Transmit and Receive 64-Byte Frame Register Definition.....	15-79
15-53	Transmit and Receive 65- to 127-Byte Frame Register Definition .....	15-80
15-54	Transmit and Received 128- to 255-Byte Frame Register Definition .....	15-80
15-55	Transmit and Received 256- to 511-Byte Frame Register Definition.....	15-81
15-56	Transmit and Received 512- to 1023-Byte Frame Register Definition .....	15-81
15-57	Transmit and Received 1024- to 1518-Byte Frame Register Definition .....	15-82
15-58	Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition .....	15-82
15-59	Receive Byte Counter Register Definition.....	15-83
15-60	Receive Packet Counter Register Definition .....	15-83
15-61	Receive FCS Error Counter Register Definition.....	15-84
15-62	Receive Multicast Packet Counter Register Definition .....	15-84
15-63	Receive Broadcast Packet Counter Register Definition .....	15-85
15-64	Receive Control Frame Packet Counter Register Definition .....	15-85
15-65	Receive Pause Frame Packet Counter Register Definition .....	15-86
15-66	Receive Unknown OPCode Packet Counter Register Definition .....	15-86
15-67	Receive Alignment Error Counter Register Definition.....	15-87
15-68	Receive Frame Length Error Counter Register Definition .....	15-87
15-69	Receive Code Error Counter Register Definition .....	15-88
15-70	Receive Carrier Sense Error Counter Register Definition .....	15-88
15-71	Receive Undersize Packet Counter Register Definition .....	15-89
15-72	Receive Oversize Packet Counter Register Definition .....	15-89
15-73	Receive Fragments Counter Register Definition .....	15-90
15-74	Receive Jabber Counter Register Definition.....	15-90
15-75	Receive Dropped Packet Counter Register Definition .....	15-91
15-76	Transmit Byte Counter Register Definition .....	15-91
15-77	Transmit Packet Counter Register Definition .....	15-92
15-78	Transmit Multicast Packet Counter Register Definition.....	15-92
15-79	Transmit Broadcast Packet Counter Register Definition.....	15-93
15-80	Transmit Pause Control Frame Counter Register Definition.....	15-93
15-81	Transmit Deferral Packet Counter Register Definition.....	15-94
15-82	Transmit Excessive Deferral Packet Counter Register Definition.....	15-94
15-83	Transmit Single Collision Packet Counter Register Definition .....	15-95

# Figures

Figure Number	Title	Page Number
15-84	Transmit Multiple Collision Packet Counter Register Definition.....	15-95
15-85	Transmit Late Collision Packet Counter Register Definition .....	15-96
15-86	Transmit Excessive Collision Packet Counter Register Definition .....	15-96
15-87	Transmit Total Collision Counter Register Definition .....	15-97
15-88	Transmit Drop Frame Counter Register Definition .....	15-97
15-89	Transmit Jabber Frame Counter Register Definition .....	15-98
15-90	Transmit FCS Error Counter Register Definition .....	15-98
15-91	Transmit Control Frame Counter Register Definition .....	15-99
15-92	Transmit Oversized Frame Counter Register Definition .....	15-99
15-93	Transmit Undersize Frame Counter Register Definition .....	15-100
15-94	Transmit Fragment Counter Register Definition .....	15-100
15-95	Carry Register 1 (CAR1) Register Definition.....	15-101
15-96	Carry Register 2 (CAR2) Register Definition.....	15-103
15-97	Carry Mask Register 1 (CAM1) Register Definition.....	15-104
15-98	Carry Mask Register 2 (CAM2) Register Definition.....	15-105
15-99	Receive Filer Rejected Packet Counter Register Definition .....	15-106
15-100	IGADDR <sub>n</sub> Register Definition .....	15-107
15-101	GADDR <sub>n</sub> Register Definition.....	15-108
15-102	FIFOCFG Register Definition .....	15-109
15-103	ATTR Register Definition .....	15-111
15-104	ATTRELI Register Definition.....	15-112
15-105	RQPRM Register Definition .....	15-113
15-106	RFBPTR0–RFBPTR7 Register Definition.....	15-113
15-107	Control Register Definition.....	15-116
15-108	Status Register Definition .....	15-117
15-109	AN Advertisement Register Definition.....	15-118
15-110	AN Link Partner Base Page Ability Register Definition .....	15-120
15-111	AN Expansion Register Definition .....	15-121
15-112	AN Next Page Transmit Register Definition .....	15-122
15-113	AN Link Partner Ability Next Page Register Definition .....	15-123
15-114	Extended Status Register Definition .....	15-123
15-115	Jitter Diagnostics Register Definition .....	15-124
15-116	TBI Control Register Definition .....	15-125
15-117	eTSEC-MII Connection .....	15-127
15-118	eTSEC-RMII Connection .....	15-128
15-119	eTSEC-GMII Connection .....	15-129
15-120	eTSEC-RGMII Connection.....	15-130
15-121	eTSEC-TBI Connection .....	15-131
15-122	eTSEC-RTBI Connection .....	15-132
15-123	eTSEC-FIFO (8-Bit) Connection.....	15-139
15-124	8-Bit GMII-Style Packet FIFO Timing.....	15-139

# Figures

Figure Number	Title	Page Number
15-125	8-Bit Encoded Packet FIFO Timing .....	15-140
15-126	Definition of Custom Preamble Sequence .....	15-146
15-127	Definition of Received Preamble Sequence.....	15-147
15-128	Ethernet Address Recognition Flowchart .....	15-148
15-129	Sample C Code for Computing eTSEC Hash Table Indices.....	15-150
15-130	Location of Frame Control Blocks for TOE Parameters .....	15-158
15-131	Transmit Frame Control Block .....	15-158
15-132	Receive Frame Control Block.....	15-160
15-133	Structure of the Receive Queue Filter Table .....	15-163
15-134	Example of eTSEC Memory Structure for BDs .....	15-174
15-135	Buffer Descriptor Ring.....	15-174
15-136	Transmit Buffer Descriptor .....	15-175
15-137	Mapping of TxBDs to a C Data Structure.....	15-175
15-138	Receive Buffer Descriptor.....	15-178
15-139	Mapping of RxBDs to a C Data Structure .....	15-179
16-1	DMA Block Diagram.....	16-1
16-2	DMA Operational Flow Chart .....	16-4
16-3	DMA Signal Summary.....	16-5
16-4	DMA Mode Registers (MR $n$ ) .....	16-10
16-5	Status Registers (SR $n$ ).....	16-12
16-6	Basic Chaining Mode Flow Chart.....	16-14
16-7	Extended Current Link Descriptor Address Registers (ECLNDAR $n$ ) .....	16-14
16-8	Current Link Descriptor Address Registers (CLNDAR $n$ ).....	16-15
16-9	Source Attributes Registers (SATR $n$ ) .....	16-16
16-10	Source Address Registers (SAR $n$ ) .....	16-17
16-11	Destination Attributes Registers (DATR $n$ ) .....	16-17
16-12	Destination Address Registers (DAR $n$ ) .....	16-19
16-13	Byte Count Registers (BCR $n$ ).....	16-19
16-14	Next Link Descriptor Address Registers (NLNDAR $n$ ) .....	16-20
16-15	Extended Next Link Descriptor Address Registers (ENLNDAR $n$ ).....	16-20
16-16	Extended Current List Descriptor Address Registers (ECLSDAR $n$ ) .....	16-21
16-17	Current List Descriptor Address Registers (CLSDAR $n$ ).....	16-21
16-18	Extended Next List Descriptor Address Registers (ENLSDAR $n$ ).....	16-22
16-19	Next List Descriptor Address Registers (NLSDAR $n$ ) .....	16-22
16-20	Source Stride Registers (SSR $n$ ) .....	16-23
16-21	Destination Stride Registers (DSR $n$ ) .....	16-23
16-22	DMA General Status Register (DGSR) .....	16-24
16-23	External Control Interface Timing .....	16-31
16-24	Stride Size and Stride Distance .....	16-33
16-25	DMA Transaction Flow with DMA Descriptors .....	16-36
16-26	List Descriptor Format .....	16-37

# Figures

Figure Number	Title	Page Number
16-27	Link Descriptor Format.....	16-37
16-28	DMA Data Paths .....	16-39
17-1	PCI Controller Block Diagram .....	17-2
17-2	PCI Interface External Signals.....	17-6
17-3	PCI CFG_ADDR Register.....	17-14
17-4	PCI CFG_DATA Register.....	17-15
17-5	PCI INT_ACK Register.....	17-15
17-6	PCI Outbound Translation Address Registers (POTAR <sub>n</sub> ).....	17-16
17-7	PCI Outbound Translation Extended Address Registers (POTEAR <sub>n</sub> ).....	17-17
17-8	PCI Outbound Window Base Address Registers (POWBAR <sub>n</sub> ).....	17-17
17-9	PCI Outbound Window 0 (Default) Attributes Register (POWAR0).....	17-18
17-10	PCI Outbound Window 1–4 Attributes Registers (POWAR1–POWAR4).....	17-18
17-11	PCI Inbound Translation Address Registers (PITAR <sub>n</sub> ).....	17-20
17-12	PCI Inbound Window Base Address Registers.....	17-21
17-13	PCI Inbound Window Base Extended Address Registers (PIWBEAR <sub>n</sub> ).....	17-21
17-14	PCI Inbound Window Attributes Registers.....	17-22
17-15	PCI Error Detect Register (ERR_DR).....	17-24
17-16	PCI Error Capture Disable Register (ERR_CAP_DR).....	17-25
17-17	PCI Error Enable Register (ERR_EN).....	17-26
17-18	PCI Error Attributes Capture Register (ERR_ATTRIB).....	17-27
17-19	PCI Error Address Capture Register (ERR_ADDR).....	17-28
17-20	PCI Error Extended Address Capture Register (ERR_EXT_ADDR).....	17-28
17-21	PCI Error Data Low Capture Register (ERR_DL).....	17-29
17-22	PCI Error Data High Capture Register (ERR_DH).....	17-29
17-23	PCI Gasket Timer Register (GAS_TIMR).....	17-29
17-24	Common PCI Configuration Header.....	17-30
17-25	PCI Vendor ID Register .....	17-30
17-26	PCI Device ID Register.....	17-31
17-27	PCI Bus Command Register .....	17-31
17-28	PCI Bus Status Register .....	17-33
17-29	PCI Revision ID Register.....	17-34
17-30	PCI Bus Programming Interface Register.....	17-34
17-31	PCI Subclass Code Register.....	17-35
17-32	PCI Bus Base Class Code Register.....	17-35
17-33	PCI Bus Cache Line Size Register.....	17-35
17-34	PCI Bus Latency Timer Register .....	17-36
17-35	PCI Configuration and Status Register Base Address Register (PCSRBAR).....	17-37
17-36	32-Bit Memory Base Address Register .....	17-37
17-37	64-Bit Low Memory Base Address Register.....	17-37
17-38	64-Bit High Memory Base Address Register .....	17-38
17-39	PCI Subsystem Vendor ID Register.....	17-38



# Figures

Figure Number	Title	Page Number
17-40	PCI Subsystem ID Register.....	17-39
17-41	PCI Bus Capabilities Pointer Register .....	17-39
17-42	PCI Bus Interrupt Line Register.....	17-39
17-43	PCI Bus Interrupt Pin Register.....	17-40
17-44	PCI Bus Minimum Grant Register (MIN_GNT) .....	17-40
17-45	PCI Bus Maximum Latency Register (MAX_LAT) .....	17-41
17-46	PCI Bus Function Register.....	17-41
17-47	PCI Bus Arbiter Configuration Register .....	17-42
17-48	PCI Arbitration Example .....	17-44
17-49	PCI Single-Beat Read Transaction.....	17-51
17-50	PCI Burst Read Transaction.....	17-51
17-51	PCI Single-Beat Write Transaction.....	17-52
17-52	PCI Burst Write Transaction .....	17-52
17-53	PCI Target-Initiated Terminations.....	17-55
17-54	DAC Single-Beat Read Example .....	17-57
17-55	DAC Burst Read Example .....	17-57
17-56	DAC Single-Beat Write Example .....	17-58
17-57	DAC Burst Write Example .....	17-58
17-58	Standard PCI Configuration Header .....	17-59
17-59	PCI Type 0 Configuration Translation.....	17-62
17-60	PCI Parity Operation.....	17-65
17-61	Address Invariant Byte Ordering—4 bytes Outbound.....	17-69
17-62	Address Invariant Byte Ordering—4 bytes Inbound .....	17-69
17-63	Address Invariant Byte Ordering—8 bytes Outbound.....	17-70
17-64	Address Invariant Byte Ordering—2 bytes Inbound .....	17-70
17-65	CFG_DATA Byte Ordering.....	17-70
18-1	PCI Express Controller Block Diagram.....	18-2
18-2	PCI Express Configuration Address Register (PEX_CONFIG_ADDR) .....	18-9
18-3	PCI Express Configuration Data Register (PEX_CONFIG_DATA).....	18-10
18-4	PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR).....	18-11
18-5	PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR).....	18-11
18-6	PCI Express Configuration Register (PEX_CONFIG).....	18-12
18-7	PCI Express PME and Message Detect Register (PEX_PME_MES_DR).....	18-13
18-8	PCI Express PME and Message Disable Register (PEX_PME_MES_DISR) .....	18-14
18-9	PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER) .....	18-16
18-10	PCI Express Power Management Command Register (PEX_PMCR) .....	18-17
18-11	IP Block Revision Register 1 .....	18-18
18-12	IP Block Revision Register 2 .....	18-19
18-13	RC Outbound Transaction Flow .....	18-20
18-14	PCI Express Outbound Translation Address Registers (PEXOTAR <sub>n</sub> ) .....	18-20

# Figures

Figure Number	Title	Page Number
18-15	PCI Express Outbound Translation Extended Address Registers (PEXOTEAR <sub>n</sub> ) .....	18-21
18-16	PCI Express Outbound Window Base Address Registers (PEXOWBAR <sub>n</sub> ) .....	18-21
18-17	PCI Express Outbound Window Attributes Register 0 (PEXOWAR0).....	18-22
18-18	PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR <sub>n</sub> ) .....	18-22
18-19	RC Inbound Transaction Flow .....	18-25
18-20	PCI Express Inbound Translation Address Registers (PEXITAR <sub>n</sub> ) .....	18-25
18-21	PCI Express Inbound Window Base Address Registers (PEXIWBAR <sub>n</sub> ).....	18-26
18-22	PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR <sub>n</sub> ) .....	18-27
18-23	PCI Express Inbound Window Attributes Registers (PEXIWAR <sub>n</sub> ).....	18-27
18-24	PCI Express Error Detect Register (PEX_ERR_DR) .....	18-30
18-25	PCI Express Error Interrupt Enable Register (PEX_ERR_EN).....	18-32
18-26	PCI Express Error Disable Register (PEX_ERR_DISR).....	18-34
18-27	PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT).....	18-35
18-28	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction.....	18-36
18-29	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction .....	18-37
18-30	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) Internal Source, Outbound Transaction.....	18-38
18-31	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) External Source, Inbound Transaction .....	18-38
18-32	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source, Outbound Transaction.....	18-39
18-33	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source, Inbound Transaction .....	18-40
18-34	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) Internal Source, Outbound Transaction.....	18-41
18-35	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) External Source, Inbound Transaction .....	18-41
18-36	PCI Express PCI-Compatible Configuration Header Common Registers.....	18-43
18-37	PCI Express Vendor ID Register.....	18-44
18-38	PCI Express Device ID Register.....	18-44
18-39	PCI Express Command Register.....	18-45
18-40	PCI Express Status Register.....	18-46
18-41	PCI Express Revision ID Register .....	18-47
18-42	PCI Express Class Code Register .....	18-48
18-43	PCI Express Bus Cache Line Size Register.....	18-48
18-44	PCI Express Bus Latency Timer Register.....	18-49
18-45	PCI Express Bus Latency Timer Register.....	18-49

# Figures

Figure Number	Title	Page Number
18-46	PCI Express PCI-Compatible Configuration Header—Type 0.....	18-50
18-47	PCI Express Base Address Register 0 (PEXCSRBAR).....	18-51
18-48	32-Bit Memory Base Address Register (BAR1).....	18-51
18-49	64-Bit Low Memory Base Address Register .....	18-52
18-50	64-Bit High Memory Base Address Register .....	18-52
18-51	PCI Express Subsystem Vendor ID Register .....	18-53
18-52	PCI Express Subsystem ID Register .....	18-53
18-53	Capabilities Pointer Register.....	18-54
18-54	PCI Express Interrupt Line Register .....	18-54
18-55	PCI Express Interrupt Pin Register .....	18-55
18-56	PCI Express Maximum Grant Register (MAX_GNT) .....	18-55
18-57	PCI Express Maximum Latency Register (MAX_LAT).....	18-56
18-58	PCI Express PCI-Compatible Configuration Header—Type 1.....	18-56
18-59	PCI Express Base Address Register 0 (PEXCSRBAR).....	18-57
18-60	PCI Express Primary Bus Number Register .....	18-57
18-61	PCI Express Secondary Bus Number Register .....	18-58
18-62	PCI Express Subordinate Bus Number Register.....	18-58
18-63	PCI Express I/O Base Register .....	18-59
18-64	PCI Express I/O Limit Register .....	18-59
18-65	PCI Express Secondary Status Register.....	18-60
18-66	PCI Express Memory Base Register.....	18-61
18-67	PCI Express Memory Limit Register.....	18-61
18-68	PCI Express Prefetchable Memory Base Register.....	18-62
18-69	PCI Express Prefetchable Memory Limit Register.....	18-62
18-70	PCI Express Prefetchable Base Upper 32 Bits Register .....	18-63
18-71	PCI Express Prefetchable Limit Upper 32 Bits Register .....	18-63
18-72	PCI Express I/O Base Upper 16 Bits Register.....	18-63
18-73	PCI Express I/O Limit Upper 16 Bits Register.....	18-64
18-74	Capabilities Pointer Register.....	18-64
18-75	PCI Express Interrupt Line Register .....	18-65
18-76	PCI Express Interrupt Pin Register .....	18-65
18-77	PCI Express Bridge Control Register .....	18-66
18-78	PCI Compatible Device-Specific Configuration Space .....	18-67
18-79	PCI Express Power Management Capability ID Register .....	18-68
18-80	PCI Express Power Management Capabilities Register .....	18-68
18-81	PCI Express Power Management Status and Control Register.....	18-69
18-82	PCI Express Power Management Data Register.....	18-69
18-83	PCI Express Capability ID Register.....	18-70
18-84	PCI Express Capabilities Register .....	18-70
18-85	PCI Express Device Capabilities Register .....	18-71
18-86	PCI Express Device Control Register.....	18-71

# Figures

Figure Number	Title	Page Number
18-87	PCI Express Device Status Register .....	18-72
18-88	PCI Express Link Capabilities Register .....	18-73
18-89	PCI Express Link Control Register .....	18-73
18-90	PCI Express Link Status Register .....	18-74
18-91	PCI Express Slot Capabilities Register .....	18-75
18-92	PCI Express Slot Control Register .....	18-76
18-93	PCI Express Slot Status Register .....	18-76
18-94	PCI Express Root Control Register .....	18-77
18-95	PCI Express Root Status Register .....	18-78
18-96	PCI Express Capability ID Register .....	18-78
18-97	PCI Express MSI Message Control Register .....	18-79
18-98	PCI Express MSI Message Address Register .....	18-79
18-99	PCI Express MSI Message Upper Address Register .....	18-79
18-100	PCI Express MSI Message Data Register .....	18-80
18-101	PCI Express Extended Configuration Space .....	18-81
18-102	PCI Express Advanced Error Reporting Capability ID Register .....	18-82
18-103	PCI Express Uncorrectable Error Status Register .....	18-82
18-104	PCI Express Uncorrectable Error Mask Register .....	18-83
18-105	PCI Express Uncorrectable Error Severity Register .....	18-84
18-106	PCI Express Correctable Error Status Register .....	18-85
18-107	PCI Express Correctable Error Mask Register .....	18-85
18-108	PCI Express Advanced Error Capabilities and Control Register .....	18-86
18-109	PCI Express Header Log Register .....	18-87
18-110	PCI Express Root Error Command Register .....	18-88
18-111	PCI Express Root Error Command Register .....	18-88
18-112	PCI Express Correctable Error Source ID Register .....	18-89
18-113	PCI Express Correctable Error Source ID Register .....	18-89
18-114	PCI Express LTSSM State Status Register (PEX_LTSSM_STAT) .....	18-90
18-115	PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO) .....	18-92
18-116	PCI Express Power Management Timer Register (PEX_PM_TIMER) .....	18-92
18-117	PCI Express PME Time-Out Register (PEX_PME_TIMEOUT) .....	18-93
18-118	PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE) .....	18-94
18-119	PCI Express Configuration Ready Register (PEX_CFG_READY) .....	18-94
18-120	PCI Express PME_To_Ack Timeout Register (PEX_PME_TO_ACK_TOR) .....	18-95
18-121	PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK) .....	18-96
18-122	Requestor/Completer Relationship .....	18-97
18-123	PCI Express High-Level Layering .....	18-97
18-124	PCI Express Packet Flow .....	18-98
18-125	Address Invariant Byte Ordering—4 bytes Outbound .....	18-99
18-126	Address Invariant Byte Ordering—4 bytes Inbound .....	18-99
18-127	Address Invariant Byte Ordering—8 bytes Outbound .....	18-100

# Figures

Figure Number	Title	Page Number
18-128	Address Invariant Byte Ordering—2 bytes Inbound .....	18-100
18-129	PEX_CONFIG_DATA Byte Ordering .....	18-100
18-130	WAKE Generation Example .....	18-108
19-1	POR PLL Status Register (PORPLLSR) .....	19-5
19-2	POR Boot Mode Status Register (PORBMSR) .....	19-6
19-3	POR I/O Impedance Status and Control Register (PORIMPSCR) .....	19-7
19-4	POR Device Status Register (PORDEVSR) .....	19-8
19-5	POR Debug Mode Status Register (PORDBGMSR) .....	19-10
19-6	POR Device Status Register 2 (PORDEVSR2) .....	19-10
19-7	POR Configuration Register (GPPORCR) .....	19-11
19-8	General-Purpose I/O Control Register (GPIOCR) .....	19-12
19-9	General-Purpose Output Data Register (GPOUTDR) .....	19-12
19-10	General-Purpose Input Data Register (GPINDR) .....	19-13
19-11	Alternate Function Pin Multiplex Control Register (PMUXCR) .....	19-13
19-12	Device Disable Register (DEVDISR) .....	19-14
19-13	Power Management Control and Status Register (POWMGTCSR) .....	19-16
19-14	Machine Check Summary Register (MCPSUMR) .....	19-17
19-15	Reset Request Status and Control Register (RSTRSCR) .....	19-18
19-16	Processor Version Register (PVR) .....	19-19
19-17	System Version Register (SVR) .....	19-19
19-18	Reset Control Register (RSTCR) .....	19-20
19-19	LBC Voltage Select Control Register (LBIUVSELCR) .....	19-20
19-20	DDR Calibration Status Register (DDRCR) .....	19-21
19-21	DDR Control Driver Register (DDRCDR) .....	19-21
19-22	DDR Clock Disable Register (DDRCLKDR) .....	19-22
19-23	Clock Out Control Register (CLKOCR) .....	19-23
19-24	SerDes 1 Control Register 1 (SRDS1CR1) .....	19-24
19-25	SerDes 2 Control Register 1 (SRDS2CR1) .....	19-25
19-26	e500 Core Power Management State Diagram .....	19-26
19-27	MPC8533E Power Management Handshaking Signals .....	19-30
20-1	Performance Monitor Block Diagram .....	20-2
20-2	Performance Monitor Global Control Register (PMGC0) .....	20-5
20-3	Performance Monitor Local Control Register A0 (PMLCA0) .....	20-6
20-4	Performance Monitor Local Control A Registers (PMLCA1–PMLCA9) .....	20-7
20-5	Performance Monitor Local Control Register B0 (PMLCB0) .....	20-8
20-6	Performance Monitor Local Control Register B (PMLCB1–PMLCB9) .....	20-9
20-7	Performance Monitor Counter Register 0 (PMC0) .....	20-10
20-8	Performance Monitor Counter Register (PMC1–PMC9) .....	20-11
20-9	Duration Threshold Event Sequence Timing Diagram .....	20-13
20-10	Burst Size, Distance, Granularity, and Burstiness Counting .....	20-14
20-11	Burstiness Counting Timing Diagram .....	20-16

# Figures

Figure Number	Title	Page Number
21-1	Debug and Watchpoint Monitor Block Diagram .....	21-2
21-2	Watchpoint Monitor Control Register 0 (WMCR0) .....	21-10
21-3	Watchpoint Monitor Control Register 1 (WMCR1) .....	21-12
21-4	Watchpoint Monitor Address Register (WMAR) .....	21-12
21-5	Watchpoint Monitor Address Mask Register (WMAMR).....	21-14
21-6	Watchpoint Monitor Transaction Mask Register (WMTMR).....	21-14
21-7	Watchpoint Monitor Status Register (WMSR) .....	21-16
21-8	Trace Buffer Control Register 0 (TBCR0).....	21-16
21-9	Trace Buffer Control Register 1 (TBCR1).....	21-18
21-10	Trace Buffer Address Register (TBAR).....	21-19
21-11	Trace Buffer Address Mask Register (TBAMR) .....	21-19
21-12	Trace Buffer Transaction Mask Register (TBTMR) .....	21-20
21-13	Trace Buffer Status Register (TBSR).....	21-20
21-14	Trace Buffer Access Control Register (TBACR)) .....	21-21
21-15	Trace Buffer Read High Register (TBADHR).....	21-22
21-16	Trace Buffer Access Data Register (TBADR).....	21-23
21-17	Programmed Context ID Register (PCIDR) .....	21-23
21-18	Current Context ID Register (CCIDR) .....	21-24
21-19	Trigger Out Source Register (TOSR).....	21-24
21-20	e500 Coherency Module Dispatch (CMD) Trace Buffer Entry .....	21-28
21-21	DDR Trace Buffer Entry .....	21-29
21-22	PCI Trace Buffer Entry .....	21-30
21-23	PCI Express Trace Buffer Entry.....	21-30

# Tables

Table Number	Title	Page Number
2-1	Target Interface Codes .....	2-1
2-2	Local Access Windows Example.....	2-2
2-3	Format of ATMU Window Definitions.....	2-3
2-4	Local Access Register Memory Map.....	2-5
2-5	LAIPBRR1 Field Descriptions .....	2-6
2-6	LAIPBRR2 Field Descriptions .....	2-6
2-7	LAWBAR <sub>n</sub> Field Descriptions .....	2-7
2-8	LAWAR <sub>n</sub> Field Descriptions .....	2-7
2-9	Overlapping Local Access Windows .....	2-8
2-10	Local Memory Configuration, Control, and Status Register Summary.....	2-11
2-11	CCSR Block Base Address Map.....	2-15
3-1	MPC8533E Signal Reference by Functional Block.....	3-5
3-2	Alphabetical Signal Reference.....	3-10
3-3	MPC8533E Reset Configuration Signals.....	3-16
3-4	Output Signal States During System Reset.....	3-17
4-1	Signal Summary .....	4-1
4-2	System Control Signals—Detailed Signal Descriptions.....	4-2
4-3	Clock Signals—Detailed Signal Descriptions .....	4-3
4-4	Local Configuration Control Register Map .....	4-4
4-5	CCSRBAR Bit Settings .....	4-5
4-6	ALTCBAR Bit Settings.....	4-6
4-7	ALTCAR Bit Settings .....	4-7
4-8	BPTR Bit Settings .....	4-8
4-9	CCB Clock PLL Ratio .....	4-12
4-10	e500 Core Clock PLL Ratios .....	4-12
4-11	SEC Mode Configuration.....	4-13
4-12	Boot ROM Location.....	4-13
4-13	Host/Agent Configuration.....	4-14
4-14	I/O Port Selection.....	4-15
4-15	CPU Boot Configuration.....	4-16
4-16	Boot Sequencer Configuration.....	4-16
4-17	DDR DRAM Type .....	4-17
4-18	eTSEC1/eTSEC2 Width Configuration .....	4-17
4-19	eTSEC3/eTSEC4 Width Configuration .....	4-18
4-20	eTSEC1 Protocol Configuration .....	4-18
4-21	eTSEC3 Protocol Configuration .....	4-19
4-22	PCI Clock Select .....	4-19
4-23	PCI Speed Configuration .....	4-19
4-24	PCI I/O Impedance.....	4-20
4-25	PCI Arbiter Configuration .....	4-20
4-26	Memory Debug Configuration.....	4-20

# Tables

Table Number	Title	Page Number
4-27	DDR Debug Configuration .....	4-21
4-28	General-Purpose POR Configuration.....	4-21
4-29	High Speed Interface Clocking .....	4-22
5-1	Device Revision Level Cross-Reference .....	5-4
5-2	Performance Monitor Instructions .....	5-11
5-3	Cache Locking Instructions .....	5-11
5-4	Scalar and Vector Embedded Floating-Point Instructions .....	5-12
5-5	BTB Locking Instructions.....	5-13
5-6	Interrupt Registers.....	5-20
5-7	Interrupt Vector Registers and Exception Conditions.....	5-21
5-8	Differences Between the e500 Core and the PowerQUICC III Core Implementation .....	5-31
6-1	Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation).....	6-4
6-2	Additional SPRs (by SPR Abbreviation).....	6-7
6-3	XER Field Description.....	6-9
6-4	BI Operand Settings for CR Fields .....	6-9
6-5	CR0 Bit Descriptions .....	6-11
6-6	MSR Field Descriptions.....	6-12
6-7	PVR Field Descriptions .....	6-14
6-8	SVR Field Descriptions .....	6-14
6-9	TCR Field Descriptions .....	6-15
6-10	TSR Field Descriptions.....	6-16
6-11	IVOR Assignments .....	6-18
6-12	ESR Field Descriptions.....	6-19
6-13	MCSR Field Descriptions .....	6-21
6-14	SPR Assignments .....	6-22
6-15	BBEAR Field Descriptions.....	6-23
6-16	BBTAR Field Descriptions .....	6-23
6-17	BUCSR Field Descriptions.....	6-24
6-18	HID0 Field Descriptions .....	6-25
6-19	HID1 Field Descriptions .....	6-26
6-20	L1CSR0 Field Descriptions .....	6-28
6-21	L1CSR1 Field Descriptions .....	6-29
6-22	L1CFG0 Field Descriptions.....	6-30
6-23	L1CFG1 Field Descriptions.....	6-31
6-24	MMUCSR0 Field Descriptions.....	6-32
6-25	MMUCFG Field Descriptions .....	6-33
6-26	TLB0CFG Field Descriptions.....	6-33
6-27	TLB1CFG Field Descriptions.....	6-34
6-28	MAS0 Field Descriptions—MMU Read/Write and Replacement Control .....	6-35
6-29	MAS1 Field Descriptions—Descriptor Context and Configuration Control.....	6-36
6-30	MAS2 Field Descriptions—EPN and Page Attributes .....	6-36



# Tables

Table Number	Title	Page Number
6-31	MAS3 Field Descriptions—RPN and Access Control .....	6-37
6-32	MAS4 Field Descriptions—Hardware Replacement Assist Configuration.....	6-38
6-33	MAS6—TLB Search Context Register 0.....	6-39
6-34	MAS 7 Field Descriptions—High Order RPN .....	6-39
6-35	DBCR0 Field Descriptions .....	6-40
6-36	DBCR1 Field Descriptions .....	6-41
6-37	DBCR2 Field Descriptions .....	6-42
6-38	DBSR Field Descriptions.....	6-44
6-39	SPEFSCR Field Descriptions.....	6-45
6-40	ACC Field Descriptions.....	6-47
6-41	Supervisor-Level PMRs (PMR[5] = 1).....	6-48
6-42	User-Level PMRs (PMR[5] = 0) (Read Only).....	6-48
6-43	PMGC0 Field Descriptions.....	6-49
6-44	PMLCa0–PMLCa3 Field Descriptions .....	6-50
6-45	PMLCb0–PMLCb3 Field Descriptions .....	6-51
6-46	PMC0–PMC3 Field Descriptions .....	6-52
1	Available L2 Cache/SRAM Configurations.....	7-3
7-2	Way Selection for SRAM Accesses.....	7-6
7-3	L2/SRAM Memory-Mapped Registers.....	7-9
7-4	L2CTL Field Descriptions .....	7-10
7-5	L2CEWAR <sub>n</sub> Field Descriptions.....	7-14
7-6	L2CEWAREA <sub>n</sub> Field Descriptions .....	7-14
7-7	L2CEWCR <sub>n</sub> Field Descriptions.....	7-15
7-8	L2SRBAR <sub>n</sub> Field Descriptions.....	7-16
7-9	L2SRBAREA <sub>n</sub> Field Descriptions .....	7-17
7-10	L2ERRINJHI Field Description.....	7-18
7-11	L2ERRINJLO Field Description .....	7-19
7-12	L2ERRINJCTL Field Descriptions.....	7-19
7-13	L2CAPTDATAHI Field Description.....	7-20
7-14	L2CAPTDATALO Field Description.....	7-20
7-15	L2CAPTECC Field Descriptions .....	7-21
7-16	L2ERRDET Field Descriptions .....	7-21
7-17	L2ERRDIS Field Descriptions.....	7-22
7-18	L2ERRINTEN Field Descriptions .....	7-23
7-19	L2ERRATTR Field Descriptions .....	7-23
7-20	L2ERRADDRH Field Description .....	7-24
7-21	L2ERRADDRL Field Description.....	7-25
7-22	L2ERRCTL Field Descriptions .....	7-25
7-23	Fastest Read Timing—Hit in L2 .....	7-27
7-24	PLRU Bit Update Algorithm .....	7-32
7-25	PLRU-Based Victim Selection Mechanism.....	7-33

# Tables

Table Number	Title	Page Number
7-26	L2 Cache States.....	7-35
7-27	State Transitions Due to Core-Initiated Transactions .....	7-35
7-28	State Transitions Due to System-Initiated Transactions .....	7-38
7-29	L2 Cache ECC Syndrome Encoding.....	7-39
7-30	L2 Cache ECC Syndrome Encoding (Check Bits) .....	7-40
8-1	ECM Memory Map.....	8-3
8-2	EEBACR Field Descriptions .....	8-4
8-3	EEBPCR Field Descriptions .....	8-4
8-4	EIPBRR1 Field Descriptions .....	8-5
8-5	EIPBRR2 Field Descriptions .....	8-6
8-6	EEDR Field Descriptions.....	8-6
8-7	EEER Field Descriptions .....	8-7
8-8	EEATR Field Descriptions.....	8-7
8-9	EELADR Field Descriptions .....	8-8
8-10	EEHADR Field Descriptions .....	8-9
9-1	DDR Memory Interface Signal Summary .....	9-3
9-2	Memory Address Signal Mappings.....	9-4
9-3	Memory Interface Signals—Detailed Signal Descriptions.....	9-5
9-4	Clock Signals—Detailed Signal Descriptions .....	9-9
9-5	DDR Memory Controller Memory Map.....	9-9
9-6	CS <sub>n</sub> _BNDS Field Descriptions.....	9-11
9-7	CS <sub>n</sub> _CONFIG Field Descriptions .....	9-12
9-8	TIMING_CFG_3 Field Descriptions .....	9-14
9-9	TIMING_CFG_0 Field Descriptions .....	9-14
9-10	TIMING_CFG_1 Field Descriptions .....	9-16
9-11	TIMING_CFG_2 Field Descriptions .....	9-19
9-12	DDR_SDRAM_CFG Field Descriptions.....	9-21
9-13	DDR_SDRAM_CFG_2 Field Descriptions.....	9-24
9-14	DDR_SDRAM_MODE Field Descriptions.....	9-25
9-15	DDR_SDRAM_MODE_2 Field Descriptions.....	9-26
9-16	DDR_SDRAM_MD_CNTL Field Descriptions.....	9-27
9-17	Settings of DDR_SDRAM_MD_CNTL Fields .....	9-28
9-18	DDR_SDRAM_INTERVAL Field Descriptions .....	9-29
9-19	DDR_DATA_INIT Field Descriptions .....	9-29
9-20	DDR_SDRAM_CLK_CNTL Field Descriptions .....	9-30
9-21	DDR_INIT_ADDR Field Descriptions .....	9-31
9-22	DDR_INIT_EXT_ADDR Field Descriptions.....	9-31
9-23	DDR_IP_REV1 Field Descriptions .....	9-32
9-24	DDR_IP_REV2 Field Descriptions .....	9-32
9-25	DATA_ERR_INJECT_HI Field Descriptions.....	9-33
9-26	DATA_ERR_INJECT_LO Field Descriptions .....	9-33

# Tables

Table Number	Title	Page Number
9-27	ERR_INJECT Field Descriptions .....	9-34
9-28	CAPTURE_DATA_HI Field Descriptions.....	9-34
9-29	CAPTURE_DATA_LO Field Descriptions.....	9-35
9-30	CAPTURE_ECC Field Descriptions .....	9-35
9-31	ERR_DETECT Field Descriptions .....	9-36
9-32	ERR_DISABLE Field Descriptions.....	9-37
9-33	ERR_INT_EN Field Descriptions .....	9-38
9-34	CAPTURE_ATTRIBUTES Field Descriptions .....	9-38
9-35	CAPTURE_ADDRESS Field Descriptions .....	9-40
9-36	CAPTURE_EXT_ADDRESS Field Descriptions .....	9-40
9-37	ERR_SBE Field Descriptions .....	9-41
9-38	Byte Lane to Data Relationship .....	9-45
9-39	Supported DDR2 SDRAM Device Configurations .....	9-46
9-40	DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled .....	9-47
9-41	DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled .....	9-48
9-42	DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled .....	9-49
9-43	DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled .....	9-50
9-44	Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Two Banks .....	9-51
9-45	Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks .....	9-52
9-46	DDR SDRAM Command Table.....	9-53
9-47	DDR SDRAM Interface Timing Intervals .....	9-54
9-48	DDR SDRAM Power-Saving Modes Refresh Configuration.....	9-62
9-49	Memory Controller–Data Beat Ordering .....	9-64
9-50	DDR SDRAM ECC Syndrome Encoding .....	9-65
9-51	DDR SDRAM ECC Syndrome Encoding (Check Bits) .....	9-67
9-52	Memory Controller Errors .....	9-68
9-53	Memory Interface Configuration Register Initialization Parameters.....	9-68
9-54	Programming Differences Between Memory Types.....	9-69
10-1	Processor Interrupts Generated Outside the Core—Types and Sources .....	10-3
10-2	e500 Core-Generated Interrupts that Cause a Wake-Up .....	10-4
10-3	Internal Interrupt Sources.....	10-6
10-4	PIC Interface Signals .....	10-7
10-5	Interrupt Signals—Detailed Signal Descriptions .....	10-8
10-6	PIC Register Address Map.....	10-9
10-7	BRR1 Field Descriptions .....	10-18
10-8	BRR2 Field Descriptions .....	10-18
10-9	FRR Field Descriptions.....	10-19
10-10	GCR Field Descriptions .....	10-19
10-11	VIR Field Descriptions .....	10-20

# Tables

Table Number	Title	Page Number
10-12	PIR Field Descriptions .....	10-21
10-13	IPIVPR <sub>n</sub> Field Descriptions .....	10-21
10-14	SVR Field Descriptions .....	10-22
10-15	TFRR Field Descriptions .....	10-23
10-16	GTCCR <sub>n</sub> Field Descriptions .....	10-23
10-17	GTBCR <sub>n</sub> Field Descriptions .....	10-24
10-18	GTVPR <sub>n</sub> Field Descriptions .....	10-24
10-19	GTDR <sub>n</sub> Field Descriptions .....	10-25
10-20	Parameters for Hourly Interrupt Timer Cascade Example .....	10-26
10-21	TCR Field Descriptions .....	10-26
10-22	ERQSR Field Descriptions .....	10-28
10-23	IRQSR0 Field Descriptions .....	10-28
10-24	IRQSR1 Field Descriptions .....	10-29
10-25	IRQSR2 Field Descriptions .....	10-29
10-26	CISR0 Field Descriptions .....	10-30
10-27	CISR1 Field Descriptions .....	10-30
10-28	CISR2 Field Descriptions .....	10-31
10-29	PM <sub>n</sub> MR0 Field Descriptions .....	10-32
10-30	PM <sub>n</sub> MR1 Field Descriptions .....	10-32
10-31	PM <sub>n</sub> MR2 Field Descriptions .....	10-33
10-32	MSGR <sub>n</sub> Field Descriptions .....	10-33
10-33	MER Field Descriptions .....	10-34
10-34	MSR Field Descriptions .....	10-35
10-35	MSIRs Field Descriptions .....	10-35
10-36	MSISR Field Descriptions .....	10-36
10-37	MSIIR Field Descriptions .....	10-36
10-38	MSIVPR <sub>n</sub> Field Descriptions .....	10-37
10-39	MSIDR <sub>n</sub> Field Descriptions .....	10-38
10-40	EIVPR <sub>n</sub> Field Descriptions .....	10-38
10-41	EIDR <sub>n</sub> Field Descriptions .....	10-39
10-42	IIVPR <sub>n</sub> Field Descriptions .....	10-40
10-43	IIDR <sub>n</sub> Field Descriptions .....	10-41
10-44	MIVPR <sub>n</sub> Field Descriptions .....	10-42
10-45	MIDR <sub>n</sub> Field Descriptions .....	10-43
10-46	Per-CPU Registers—Private Access Address Offsets .....	10-43
10-47	IPIDR <sub>n</sub> Field Descriptions .....	10-45
10-48	CTPR Field Descriptions .....	10-45
10-49	WHOAMI Field Descriptions .....	10-46
10-50	IACK Field Descriptions .....	10-47
10-51	EOI Field Descriptions .....	10-47
10-52	PCI Express INTx/IRQ <sub>n</sub> Sharing .....	10-51

# Tables

Table Number	Title	Page Number
11-1	I <sup>2</sup> C Interface Signal Descriptions .....	11-3
11-2	I <sup>2</sup> C Interface Signal—Detailed Signal Descriptions.....	11-4
11-3	I <sup>2</sup> C Memory Map.....	11-5
11-4	I2CADR Field Descriptions.....	11-6
11-5	I2CFDR Field Descriptions .....	11-7
11-6	I2CCR Field Descriptions.....	11-8
11-7	I2CSR Field Descriptions .....	11-9
11-8	I2CDR Field Descriptions.....	11-10
11-9	I2CDFSRR Field Descriptions.....	11-11
12-1	Example Descriptor.....	12-4
12-2	SEC Base Address Map .....	12-10
12-3	SEC Address Map.....	12-11
12-4	Header Dword Bit Definitions .....	12-17
12-5	EU_SEL0 and EU_SEL1 Values .....	12-18
12-6	Descriptor Types .....	12-19
12-7	Pointer Dword Field Definitions.....	12-20
12-8	Link Table Field Definitions .....	12-22
12-9	Descriptor Format by Type .....	12-24
12-10	PKEU[ROUTINE] Field Values .....	12-27
12-11	PKEU Reset Control Register Field Descriptions .....	12-29
12-12	PKEU Status Register Field Descriptions.....	12-30
12-13	PKEU interrupt Status Register Field Descriptions.....	12-31
12-14	PKEU Interrupt Control Register Field Descriptions .....	12-32
12-15	DEU Mode Register Field Descriptions .....	12-34
12-16	DEU Key Size Register Field Descriptions .....	12-34
12-17	DEU Reset Control Register Field Descriptions.....	12-35
12-18	DEU Status Register Field Descriptions.....	12-36
12-19	DEU Interrupt Status Register Field Descriptions.....	12-37
12-20	DEU Interrupt Control Register Field Descriptions .....	12-39
12-21	AFEU Mode Register Field Descriptions .....	12-43
12-22	AFEU Reset Control Register Field Descriptions .....	12-45
12-23	AFEU Status Register Field Descriptions.....	12-46
12-24	AFEU Interrupt Status Register Field Descriptions.....	12-47
12-25	AFEU Interrupt Control Register Field Descriptions .....	12-48
12-26	MDEU Mode Register in Old Configuration (NEW = 0).....	12-51
12-27	MDEU Mode Register in New Configuration (NEW = 1) .....	12-52
12-28	Mode Register —HMAC or SSL-MAC Generated by Single Descriptor.....	12-53
12-29	Mode Register —HMAC Generated Across a Sequence of Descriptors.....	12-54
12-30	MDEU Reset Control Register Field Descriptions.....	12-55
12-31	MDEU Status Register Field Descriptions .....	12-56
12-32	MDEU Interrupt Status Register Field Descriptions .....	12-57

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
12-33	MDEU Interrupt Control Register Field Descriptions .....	12-58
12-34	RNG Reset Control Register Field Descriptions .....	12-64
12-35	RNG Status Register Field Descriptions .....	12-64
12-36	RNG Interrupt Status Register Field Descriptions .....	12-65
12-37	RNG Interrupt Control Register Field Descriptions .....	12-66
12-38	AESU Mode Register .....	12-68
12-39	AES Cipher Modes .....	12-68
12-40	AESU Reset Control Register Field Descriptions .....	12-70
12-41	AESU Status Register Field Descriptions .....	12-71
12-42	AESU Interrupt Status Register Field Descriptions .....	12-72
12-43	AESU Interrupt Control Register Field Descriptions .....	12-73
12-44	KEU Mode Register Field Descriptions .....	12-80
12-45	KEU Reset Control Register Field Descriptions .....	12-82
12-46	KEU Status Register Field Descriptions .....	12-83
12-47	KEU Interrupt Status Register Field Descriptions .....	12-84
12-48	KEU Interrupt Control Register Field Descriptions .....	12-86
12-49	KEU IV_1 Register Field Descriptions .....	12-88
12-50	Crypto-Channel Configuration Register (CCCR) Field Descriptions .....	12-93
12-51	Header Dword Writeback Field Descriptions .....	12-94
12-52	Crypto-Channel Pointer Status Register Field Descriptions .....	12-95
12-53	G_STATE and S_STATE Field Values .....	12-97
12-54	CHN_STATE Field Values .....	12-98
12-55	Crypto-Channel Pointer Status Register Error Field Descriptions .....	12-98
12-56	Channel Pointer Status Register PTR_DW Field Values .....	12-99
12-57	Channel Current Descriptor Pointer Register Field Descriptions .....	12-100
12-58	Fetch FIFO Field Descriptions .....	12-101
12-59	Channel Assignment Value .....	12-109
12-60	Field Names in Interrupt Mask, Interrupt Status, and Interrupt Clear Registers .....	12-110
12-61	Channel Current Descriptor Pointer Register Signals .....	12-113
12-62	Master Control Register (MCR) Field Descriptions .....	12-114
13-1	DUART Signal Overview .....	13-3
13-2	DUART Signals—Detailed Signal Descriptions .....	13-3
13-3	DUART Register Summary .....	13-5
13-4	URBR Field Descriptions .....	13-6
13-5	UTHR Field Descriptions .....	13-7
13-6	UDMB Field Descriptions .....	13-7
13-7	UDLB Field Descriptions .....	13-8
13-8	Baud Rate Examples .....	13-8
13-9	UIER Field Descriptions .....	13-9
13-10	UIIR Field Descriptions .....	13-10
13-11	UIIR IID Bits Summary .....	13-10

# Tables

Table Number	Title	Page Number
13-12	UFCR Field Descriptions.....	13-12
13-13	ULCR Field Descriptions.....	13-13
13-14	Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS].....	13-14
13-15	UMCR Field Descriptions.....	13-14
13-16	ULSR Field Descriptions.....	13-15
13-17	UMSR Field Descriptions.....	13-16
13-18	USCR Field Descriptions.....	13-17
13-19	UAFR Field Descriptions.....	13-18
13-20	UDSR Field Descriptions.....	13-18
13-21	UDSR[TXRDY] Set Conditions.....	13-19
13-22	UDSR[TXRDY] Cleared Conditions.....	13-19
13-23	UDSR[RXRDY] Set Conditions.....	13-19
13-24	UDSR[RXRDY] Cleared.....	13-19
14-1	Signal Properties—Summary.....	14-4
14-2	Local Bus Controller Detailed Signal Descriptions.....	14-5
14-3	Local Bus Controller Memory Map.....	14-9
14-4	BR <sub>n</sub> Field Descriptions.....	14-11
14-5	Memory Bank Sizes in Relation to Address Mask.....	14-12
14-6	OR <sub>n</sub> —GPCM Field Descriptions.....	14-13
14-7	OR <sub>n</sub> —UPM Field Descriptions.....	14-15
14-8	OR <sub>n</sub> —SDRAM Field Descriptions.....	14-16
14-9	MAR Field Descriptions.....	14-17
14-10	MxMR Field Descriptions.....	14-17
14-11	MRTPR Field Descriptions.....	14-20
14-12	MDR Field Descriptions.....	14-20
14-13	LSDMR Field Descriptions.....	14-21
14-14	LURT Field Descriptions.....	14-23
14-15	LSRT Field Descriptions.....	14-24
14-16	LTESR Field Descriptions.....	14-25
14-17	LTEDR Field Descriptions.....	14-26
14-18	LTEIR Field Descriptions.....	14-27
14-19	LTEATR Field Descriptions.....	14-28
14-20	LTEAR Field Descriptions.....	14-29
14-21	LBCR Field Descriptions.....	14-29
14-22	LCRR Field Descriptions.....	14-31
14-23	GPCM Write Control Signal Timing.....	14-38
14-24	GPCM Read Control Signal Timing.....	14-39
14-25	Boot Bank Field Values After Reset.....	14-47
14-26	SDRAM Interface Commands.....	14-49
14-27	UPM Routines Start Addresses.....	14-60
14-28	RAM Word Field Descriptions.....	14-65

# Tables

Table Number	Title	Page Number
14-29	MxMR Loop Field Use .....	14-69
14-30	UPM Address Multiplexing .....	14-70
14-31	Data Bus Requirements For Read Cycle.....	14-84
14-32	Typical SDRAM Devices.....	14-86
14-33	LAD <sub>n</sub> Signal Connections to 128-Mbyte SDRAM.....	14-88
14-34	Logical Address Bus Partitioning .....	14-89
14-35	SDRAM Device Address Port During Address Phase.....	14-89
14-36	SDRAM Device Address Port During READ/WRITE Command.....	14-89
14-37	Register Settings for 128-Mbyte SDRAMs .....	14-90
14-38	Logical Address Partitioning .....	14-90
14-39	SDRAM Device Address Port During Address Phase.....	14-91
14-40	SDRAM Device Address Port During READ/WRITE Command.....	14-91
14-41	Register Settings for 512-Mbyte SDRAMs .....	14-91
14-42	SDRAM Capacitance .....	14-93
14-43	SDRAM AC Characteristics .....	14-94
14-44	Local Bus to MSC8101 HDI16 Connections.....	14-99
14-45	UPM Synchronization Cycles .....	14-107
15-1	eTSEC <sub>n</sub> Network Interface Signal Properties .....	15-7
15-2	eTSEC Signals—Detailed Signal Descriptions .....	15-9
15-3	Module Memory Map Summary.....	15-13
15-4	Module Memory Map .....	15-14
15-5	TSEC_ID Field Descriptions .....	15-23
15-6	TSEC_ID2 Field Descriptions .....	15-24
15-7	IEVENT Field Descriptions.....	15-25
15-8	IMASK Field Descriptions .....	15-28
15-9	EDIS Field Descriptions .....	15-30
15-10	ECNTRL Field Descriptions.....	15-32
15-11	eTSEC Interface Configurations.....	15-33
15-12	PTV Field Descriptions.....	15-34
15-13	DMACTRL Field Descriptions.....	15-34
15-14	TBIPA Field Descriptions .....	15-36
15-15	TCTRL Field Descriptions.....	15-36
15-16	TSTAT Field Descriptions.....	15-39
15-17	DFVLAN Field Descriptions .....	15-42
15-18	TXIC Field Descriptions .....	15-43
15-19	TQUEUE Field Descriptions .....	15-44
15-20	TR03WT Field Descriptions.....	15-45
15-21	TR47WT Field Descriptions .....	15-46
15-22	TBDBPH Field Descriptions .....	15-46
15-23	TBPTR <sub>n</sub> Field Descriptions .....	15-47
15-24	TBASEH Field Descriptions.....	15-47



# Tables

Table Number	Title	Page Number
15-25	TBASE0–TBASE7 Field Descriptions .....	15-48
15-26	RCTRL Field Descriptions .....	15-49
15-27	RSTAT Field Descriptions .....	15-51
15-28	RXIC Field Descriptions.....	15-53
15-29	RQUEUE Field Descriptions .....	15-53
15-30	RBIFX Field Descriptions .....	15-55
15-31	RQFAR Field Descriptions .....	15-57
15-32	RQFCR Field Descriptions .....	15-57
15-33	RQFPR Field Descriptions.....	15-59
15-34	MRBLR Field Descriptions .....	15-61
15-35	RBDBPH Field Descriptions .....	15-62
15-36	BPTR <sub>n</sub> Field Descriptions.....	15-62
15-37	RBASEH Field Descriptions .....	15-63
15-38	RBASE0–RBASE7 Field Descriptions .....	15-63
15-39	MACCFG1 Field Descriptions .....	15-67
15-40	MACCFG2 Field Descriptions .....	15-68
15-41	IPGIFG Field Descriptions .....	15-70
15-42	HAFDUP Field Descriptions .....	15-71
15-43	MAXFRM Descriptions.....	15-72
15-44	MIIMCFG Field Descriptions.....	15-73
15-45	MIIMCOM Descriptions.....	15-73
15-46	MIIMADD Field Descriptions.....	15-74
15-47	MIIMCON Field Descriptions .....	15-75
15-48	MIIMSTAT Field Descriptions .....	15-75
15-49	MIIMIND Field Descriptions .....	15-76
15-50	IFSTAT Field Descriptions .....	15-76
15-51	MACSTNADDR1 Field Descriptions .....	15-77
15-52	MACSTNADDR2 Field Descriptions .....	15-77
15-53	MAC <sub>n</sub> ADDR1 Field Descriptions.....	15-78
15-54	MAC01ADDR2–MAC15ADDR2 Field Descriptions .....	15-79
15-55	TR64 Field Descriptions .....	15-80
15-56	TR127 Field Descriptions .....	15-80
15-57	TR255 Field Descriptions .....	15-80
15-58	TR511 Field Descriptions .....	15-81
15-59	TR1K Field Descriptions .....	15-81
15-60	TRMAX Field Descriptions.....	15-82
15-61	TRMGV Field Descriptions.....	15-82
15-62	RBYT Field Descriptions.....	15-83
15-63	RPKT Field Descriptions .....	15-83
15-64	RFCS Field Descriptions .....	15-84
15-65	RMCA Field Descriptions .....	15-84

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
15-66	RBCA Field Descriptions .....	15-85
15-67	RXCF Field Descriptions .....	15-85
15-68	RXPF Field Descriptions .....	15-86
15-69	RXUO Field Descriptions .....	15-86
15-70	RALN Field Descriptions .....	15-87
15-71	RFLR Field Descriptions .....	15-87
15-72	RCDE Field Descriptions .....	15-88
15-73	RCSE Field Descriptions .....	15-88
15-74	RUND Field Descriptions .....	15-89
15-75	ROVR Field Descriptions .....	15-89
15-76	RFRG Field Descriptions .....	15-90
15-77	RJBR Field Descriptions .....	15-90
15-78	RDRP Field Descriptions .....	15-91
15-79	TBYT Field Descriptions .....	15-91
15-80	TPKT Field Descriptions .....	15-92
15-81	TMCA Field Descriptions .....	15-92
15-82	TBCA Field Descriptions .....	15-93
15-83	TXPF Field Descriptions .....	15-93
15-84	TDFR Field Descriptions .....	15-94
15-85	TEDF Field Descriptions .....	15-94
15-86	TSCL Field Descriptions .....	15-95
15-87	TMCL Field Descriptions .....	15-95
15-88	TLCL Field Descriptions .....	15-96
15-89	TXCL Field Descriptions .....	15-96
15-90	TNCL Field Descriptions .....	15-97
15-91	TDRP Field Descriptions .....	15-97
15-92	TJBR Field Descriptions .....	15-98
15-93	TFCS Field Descriptions .....	15-98
15-94	TXCF Field Descriptions .....	15-99
15-95	TOVR Field Descriptions .....	15-99
15-96	TUND Field Descriptions .....	15-100
15-97	TFRG Field Descriptions .....	15-100
15-98	CAR1 Field Descriptions .....	15-102
15-99	CAR2 Field Descriptions .....	15-103
15-100	CAM1 Field Descriptions .....	15-104
15-101	CAM2 Field Descriptions .....	15-106
15-102	RREJ Field Descriptions .....	15-107
15-103	IGADDR <sub>n</sub> Field Descriptions .....	15-108
15-104	GADDR <sub>n</sub> Field Descriptions .....	15-108
15-105	FIFOCFG Field Descriptions .....	15-109
15-106	ATTR Field Descriptions .....	15-111

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
15-107	ATTRELI Field Descriptions .....	15-112
15-108	RQPRM Field Descriptions .....	15-113
15-109	RFBPTR0–RFBPTR7 Field Descriptions .....	15-114
15-110	TBI MII Register Set.....	15-115
15-111	CR Field Descriptions .....	15-116
15-112	SR Descriptions.....	15-117
15-113	ANA Field Descriptions.....	15-118
15-114	PAUSE Priority Resolution .....	15-119
15-115	ANLPBPA Field Descriptions .....	15-120
15-116	ANEX Field Descriptions .....	15-122
15-117	ANNPT Field Descriptions .....	15-122
15-118	ANLPANP Field Descriptions .....	15-123
15-119	EXST Field Descriptions .....	15-124
15-120	JD Field Descriptions .....	15-125
15-121	TBICON Field Descriptions .....	15-126
15-122	GMII, MII, and RMII Signals Multiplexing .....	15-133
15-123	RGMII, TBI, and RTBI Signals Multiplexing .....	15-134
15-124	RGMII and RTBI Signals Multiplexing.....	15-135
15-125	RGMII Signals Multiplexing .....	15-136
15-126	Shared Signals.....	15-137
15-127	Port Mode Combinations Supported by eTSECs.....	15-138
15-128	Signal Encoding for GMII-Style 8-Bit FIFO .....	15-140
15-129	Signal Encoding for Encoded 8-Bit FIFO.....	15-140
15-130	Steps for Minimum Register Initialization.....	15-141
15-131	Custom Preamble Field Descriptions.....	15-146
15-132	Received Preamble Field Descriptions .....	15-147
15-133	Flow Control Frame Structure .....	15-151
15-134	Non-Error Transmit Interrupts .....	15-153
15-135	Non-Error Receive Interrupts.....	15-153
15-136	Interrupt Coalescing Timing Threshold Ranges .....	15-154
15-137	Transmission Errors .....	15-155
15-138	Reception Errors .....	15-156
15-139	Tx Frame Control Block Descriptions .....	15-159
15-140	Rx Frame Control Block Descriptions.....	15-161
15-141	Special Filer Rules .....	15-165
15-142	Receive Queue Filer Interrupt Events.....	15-165
15-143	Filer Table Example—802.1p Priority Filing .....	15-166
15-144	Filer Table Example—IP Diff-Serv Code Points Filing .....	15-167
15-145	Filer Table Example—TCP and UDP Port Filing.....	15-167
15-146	Transmit Data Buffer Descriptor (TxBD) Field Descriptions .....	15-176
15-147	Receive Buffer Descriptor Field Descriptions .....	15-179

# Tables

Table Number	Title	Page Number
15-148	MII Interface Mode Signal Configuration .....	15-181
15-149	Shared MII Signals.....	15-182
15-150	MII Mode Register Initialization Steps.....	15-182
15-151	GMII Interface Mode Signal Configuration .....	15-185
15-152	Shared GMII Signals.....	15-186
15-153	GMII Mode Register Initialization Steps.....	15-186
15-154	TBI Interface Mode Signal Configuration .....	15-189
15-155	Shared TBI Signals .....	15-190
15-156	TBI Mode Register Initialization Steps.....	15-190
15-157	RGMII Interface Mode Signal Configuration.....	15-193
15-158	Shared RGMII Signals .....	15-194
15-159	RGMII Mode Register Initialization Steps .....	15-194
15-160	RMII Interface Mode Signal Configuration.....	15-197
15-161	Shared RMII Signals .....	15-198
15-162	RMII Mode Register Initialization Steps .....	15-198
15-163	RTBI Interface Mode Signal Configuration.....	15-201
15-164	Shared RTBI Signals .....	15-202
15-165	RTBI Mode Register Initialization Steps .....	15-202
15-166	8-Bit FIFO Interface Mode Signal Configurations .....	15-205
15-167	8-Bit FIFO Mode Register Initialization Steps .....	15-206
16-1	Relationship of Modes and Features .....	16-3
16-2	DMA Mode Bit Settings .....	16-3
16-3	DMA Signals—Detailed Signal Descriptions.....	16-6
16-4	DMA Register Summary .....	16-7
16-5	MR <sub>n</sub> Field Descriptions .....	16-10
16-6	SR <sub>n</sub> Field Descriptions .....	16-13
16-7	ECLNDAR <sub>n</sub> Field Descriptions .....	16-15
16-8	CLNDAR <sub>n</sub> Field Descriptions.....	16-15
16-9	SATR <sub>n</sub> Field Descriptions .....	16-16
16-10	SAR <sub>n</sub> Field Descriptions .....	16-17
16-11	DATR <sub>n</sub> Field Descriptions.....	16-18
16-12	DAR <sub>n</sub> Field Descriptions.....	16-19
16-13	BCR <sub>n</sub> Field Descriptions .....	16-19
16-14	NLNDAR <sub>n</sub> Field Descriptions.....	16-20
16-15	ENLNDAR <sub>n</sub> Field Descriptions .....	16-20
16-16	CLSDAR <sub>n</sub> Field Descriptions .....	16-21
16-17	NLSDAR <sub>n</sub> Field Descriptions .....	16-22
16-18	SSR <sub>n</sub> Field Descriptions .....	16-23
16-19	DSR <sub>n</sub> Field Descriptions .....	16-23
16-20	DGSR Field Descriptions.....	16-24
16-21	Channel State Table.....	16-32

# Tables

Table Number	Title	Page Number
16-22	List DMA Descriptor Summary.....	16-34
16-23	Link DMA Descriptor Summary .....	16-35
17-1	POR Parameters for PCI Controller.....	17-5
17-2	PCI Interface Signals—Detailed Signal Descriptions .....	17-6
17-3	PCI Memory-Mapped Register Map.....	17-11
17-4	PCI CFG_ADDR Field Descriptions.....	17-14
17-5	PCI CFG_DATA Field Descriptions.....	17-15
17-6	PCI INT_ACK Field Descriptions.....	17-15
17-7	POTAR <sub>n</sub> Field Descriptions .....	17-16
17-8	POTEAR <sub>n</sub> Field Descriptions.....	17-17
17-9	POWBAR <sub>n</sub> Field Descriptions .....	17-17
17-10	POWAR <sub>n</sub> Field Descriptions .....	17-18
17-11	PITAR <sub>n</sub> Field Descriptions .....	17-20
17-12	PIWBAR Field Descriptions.....	17-21
17-13	PIWBEAR Field Descriptions .....	17-21
17-14	PIWAR <sub>n</sub> Field Descriptions.....	17-22
17-15	ERR_DR Field Descriptions.....	17-24
17-16	ERR_CAP_DR Field Descriptions .....	17-25
17-17	ERR_EN Field Descriptions .....	17-27
17-18	ERR_ATTRIB Field Descriptions .....	17-27
17-19	ERR_ADDR Field Descriptions .....	17-28
17-20	ERR_EXT_ADDR Field Descriptions .....	17-28
17-21	ERR_DL Field Description.....	17-29
17-22	ERR_DH Field Description .....	17-29
17-23	GAS_TIMR Field Descriptions .....	17-29
17-24	PCI Vendor ID Register Field Description .....	17-31
17-25	PCI Device ID Register Field Description.....	17-31
17-26	PCI Bus Command Register Field Descriptions.....	17-31
17-27	PCI Bus Status Register Field Descriptions.....	17-33
17-28	PCI Revision ID Register Field Descriptions .....	17-34
17-29	PCI Bus Programming Interface Register Field Description.....	17-34
17-30	PCI Subclass Code Register Field Description.....	17-35
17-31	PCI Bus Base Class Code Register Field Description .....	17-35
17-32	PCI Bus Cache Line Size Register Field Descriptions .....	17-36
17-33	PCI Bus Latency Timer Register Field Descriptions.....	17-36
17-34	PCSRBAR Field Descriptions .....	17-37
17-35	32-Bit Memory Base Address Register Field Descriptions.....	17-37
17-36	64-Bit Low Memory Base Address Register Field Descriptions.....	17-38
17-37	Bit Setting for 64-Bit High Memory Base Address Register.....	17-38
17-38	PCI Subsystem Vendor ID Register Field Description .....	17-38
17-39	PCI Subsystem ID Register Field Description.....	17-39

# Tables

Table Number	Title	Page Number
17-40	PCI Bus Capabilities Pointer Register Field Description .....	17-39
17-41	PCI Bus Interrupt Line Register Field Description.....	17-40
17-42	PCI Bus Interrupt Pin Register Field Description.....	17-40
17-43	PCI Bus Minimum Grant Register Field Description .....	17-40
17-44	PCI Bus Maximum Latency Register Field Description .....	17-41
17-45	PCI Bus Function Register Field Descriptions .....	17-41
17-46	PCI Bus Arbiter Configuration Register Field Descriptions .....	17-42
17-47	PCI Bus Commands .....	17-46
17-48	Supported Combinations of PCI_AD[1:0].....	17-48
17-49	PCI Configuration Space Header Summary .....	17-59
17-50	PCI Type 0 Configuration—Device Number to AD <sub>n</sub> Translation.....	17-62
17-51	Special-Cycle Message Encodings .....	17-64
17-52	PCI Mode Error Actions .....	17-66
17-53	Affected Configuration Register Bits for POR .....	17-67
17-54	Power-On Reset Values for Affected Configuration Bits .....	17-68
18-1	POR Parameters for PCI Express Controller .....	18-4
18-2	PCI Express Interface Signals—Detailed Signal Descriptions .....	18-5
18-3	PCI Express Memory-Mapped Register Map .....	18-6
18-4	PEX_CONFIG_ADDR Field Descriptions .....	18-10
18-5	PEX_CONFIG_DATA Field Descriptions .....	18-10
18-6	PEX_OTB_CPL_TOR Field Descriptions .....	18-11
18-7	PEX_CONF_RTU_TOR Field Descriptions .....	18-12
18-8	PEX_CONFIG Field Descriptions.....	18-12
18-9	PEX_PME_MES_DR Field Descriptions.....	18-13
18-10	PEX_PME_MES_DISR Field Descriptions .....	18-15
18-11	PEX_PME_MES_IER Field Descriptions.....	18-16
18-12	PEX_PMCR Field Descriptions.....	18-18
18-13	PCI Express IP Block Revision Register 1 Field Descriptions.....	18-18
18-14	PCI Express IP Block Revision Register 2 Field Descriptions.....	18-19
18-15	PEXOTAR <sub>n</sub> Field Descriptions .....	18-20
18-16	PCI Express Outbound Extended Address Translation Register <i>n</i> Field Descriptions .....	18-21
18-17	PCI Express Outbound Window Base Address Register <i>n</i> Field Descriptions.....	18-22
18-18	PEXOWAR <sub>n</sub> Field Descriptions .....	18-22
18-19	PCI Express Inbound Translation Address Registers Field Descriptions .....	18-26
18-20	PCI Express Inbound Window Base Address Register Field Descriptions .....	18-26
18-21	PCI Express Inbound Window Base Extended Address Register Field Descriptions .....	18-27
18-22	PCI Express Inbound Window Attributes Registers Field Descriptions.....	18-27
18-23	PCI Express Error Detect Register Field Descriptions .....	18-30
18-24	PCI Express Error Interrupt Enable Register Field Descriptions .....	18-32

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
18-25	PCI Express Error Disable Register Field Descriptions .....	18-34
18-26	PCI Express Error Capture Status Register Field Descriptions .....	18-36
18-27	PCI Express Error Capture Register 0 Field Descriptions Internal Source, Outbound Transaction.....	18-37
18-28	PCI Express Error Capture Register 0 Field Descriptions External Source, Inbound Transaction .....	18-37
18-29	PCI Express Error Capture Register 1 Field Descriptions Internal Source, Outbound Transaction.....	18-38
18-30	PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound Transaction .....	18-39
18-31	PCI Express Error Capture Register 2 Field Descriptions Internal Source, Outbound Transaction.....	18-40
18-32	PCI Express Error Capture Register 2 Field Descriptions External Source, Inbound Transaction .....	18-40
18-33	PCI Express Error Capture Register 3 Field Descriptions Internal Source, Outbound Transaction.....	18-41
18-34	PEX Error Capture Register 3 Field Descriptions External Source, Inbound Transaction .....	18-41
18-35	PCI Express Vendor ID Register Field Description.....	18-44
18-36	PCI Express Device ID Register Field Description .....	18-44
18-37	PCI Express Command Register Field Descriptions .....	18-45
18-38	PCI Express Status Register Field Descriptions .....	18-46
18-39	PCI Express Revision ID Register Field Descriptions.....	18-47
18-40	PCI Express Class Code Register Field Descriptions.....	18-48
18-41	PCI Express Bus Cache Line Size Register Field Descriptions.....	18-48
18-42	PCI Express Bus Latency Timer Register Field Descriptions .....	18-49
18-43	PCI Express Bus Latency Timer Register Field Descriptions .....	18-49
18-44	PEXCSRBAR Field Descriptions .....	18-51
18-45	32-Bit Memory Base Address Register (BAR1) Field Descriptions .....	18-51
18-46	64-Bit Low Memory Base Address Register Field Descriptions.....	18-52
18-47	Bit Setting for 64-Bit High Memory Base Address Register.....	18-52
18-48	PCI Express Subsystem Vendor ID Register Field Description .....	18-53
18-49	PCI Express Subsystem ID Register Field Description .....	18-53
18-50	Capabilities Pointer Register Field Description .....	18-54
18-51	PCI Express Interrupt Line Register Field Description .....	18-54
18-52	PCI Express Interrupt Pin Register Field Description .....	18-55
18-53	PCI Express Maximum Grant Register Field Description.....	18-55
18-54	PCI Express Maximum Latency Register Field Description .....	18-56
18-55	PEXCSRBAR Field Descriptions .....	18-57
18-56	PCI Express Primary Bus Number Register Field Description .....	18-57
18-57	PCI Express Secondary Bus Number Register Field Description .....	18-58

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
18-58	PCI Express Subordinate Bus Number Register Field Description .....	18-58
18-59	PCI Express I/O Base Register Field Description .....	18-59
18-60	PCI Express I/O Limit Register Field Description .....	18-60
18-61	PCI Express Secondary Status Register Field Description .....	18-60
18-62	PCI Express Memory Base Register Field Description .....	18-61
18-63	PCI Express Memory Limit Register Field Description .....	18-61
18-64	PCI Express Prefetchable Memory Base Register Field Description .....	18-62
18-65	PCI Express Prefetchable Memory Limit Register Field Description .....	18-62
18-66	PCI Express Prefetchable Base Upper 32 Bits Register .....	18-63
18-67	PCI Express Prefetchable Limit Upper 32 Bits Register .....	18-63
18-68	PCI Express I/O Base Upper 16 Bits Register Field Description .....	18-64
18-69	PCI Express I/O Limit Upper 16 Bits Register Field Description .....	18-64
18-70	Capabilities Pointer Register Field Description .....	18-64
18-71	PCI Express Interrupt Line Register Field Description .....	18-65
18-72	PCI Express Interrupt Pin Register Field Description .....	18-65
18-73	PCI Express Bridge Control Register Field Description .....	18-66
18-74	PCI Express Power Management Capability ID Register Field Description .....	18-68
18-75	PCI Express Power Management Capabilities Register Field Description .....	18-68
18-76	PCI Express Status and Control Register Field Description .....	18-69
18-77	PCI Express Power Management Data Register Field Description .....	18-69
18-78	PCI Express Capability ID Register Field Description .....	18-70
18-79	PCI Express Capabilities Register Field Description .....	18-70
18-80	PCI Express Device Capabilities Register Field Description .....	18-71
18-81	PCI Express Device Control Register Field Description .....	18-72
18-82	PCI Express Device Status Register Field Description .....	18-72
18-83	PCI Express Link Capabilities Register Field Description .....	18-73
18-84	PCI Express Link Control Register Field Description .....	18-73
18-85	PCI Express Link Status Register Field Description .....	18-74
18-86	PCI Express Slot Capabilities Register Field Description .....	18-75
18-87	PCI Express Slot Control Register Field Description .....	18-76
18-88	PCI Express Slot Status Register Field Descriptions .....	18-76
18-89	PCI Express Root Control Register Field Description .....	18-77
18-90	PCI Express Root Status Register Field Description .....	18-78
18-91	PCI Express Capability ID Register Field Description .....	18-78
18-92	PCI Express MSI Message Control Register Field Description .....	18-79
18-93	PCI Express MSI Message Address Register Field Description .....	18-79
18-94	PCI Express MSI Message Upper Address Register Field Description .....	18-80
18-95	PCI Express MSI Message Data Register Field Description .....	18-80
18-96	PCI Express Advanced Error Reporting Capability ID Register Field Description .....	18-82
18-97	PCI Express Uncorrectable Error Status Register Field Description .....	18-82



# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
18-98	PCI Express Uncorrectable Error Mask Register Field Description.....	18-83
18-99	PCI Express Uncorrectable Error Severity Register Field Description .....	18-84
18-100	PCI Express Correctable Error Status Register Field Description.....	18-85
18-101	PCI Express Correctable Error Mask Register Field Description.....	18-86
18-102	PCI Express Advanced Error Capabilities and Control Register Field Description.....	18-86
18-103	PCI Express Header Log Register Field Description.....	18-87
18-104	PCI Express Root Error Command Register Field Description.....	18-88
18-105	PCI Express Root Error Command Register Field Description.....	18-88
18-106	PCI Express Correctable Error Source ID Register Field Description .....	18-89
18-107	PCI Express Correctable Error Source ID Register Field Description .....	18-89
18-108	PEX_LTSSM_STAT Field Descriptions .....	18-90
18-109	PEX_LTSSM_STAT Status Codes.....	18-90
18-110	PEX_GCLK_RATIO Field Descriptions .....	18-92
18-111	PEX_PM_TIMER Field Descriptions .....	18-93
18-112	PEX_PME_TIMEOUT Field Descriptions .....	18-93
18-113	PEX_SSVID_UPDATE Field Descriptions.....	18-94
18-114	PEX_CFG_READY Field Descriptions .....	18-95
18-115	PEX_PME_TO_ACK_TOR Field Descriptions.....	18-95
18-116	PEX_SS_INTR_MASK Field Descriptions .....	18-96
18-117	PCI Express Transactions .....	18-98
18-118	Internal Platform (OCeaN) Message Data Format .....	18-102
18-119	PCI Express ATMU Outbound Messages.....	18-103
18-120	PCI Express RC Inbound Message Handling .....	18-104
18-121	PCI Express EP Inbound Message Handling.....	18-105
18-122	Initial Credit Advertisement.....	18-107
18-123	Power Management State Supported .....	18-108
19-1	External Signal Summary .....	19-2
19-2	Detailed Signal Descriptions.....	19-2
19-3	Global Utilities Block Register Summary .....	19-3
19-4	PORPLLSR Field Descriptions .....	19-5
19-5	PORBMSR Field Descriptions .....	19-6
19-6	PORIMPSCR Field Descriptions.....	19-7
19-7	PORDEVSR Field Descriptions .....	19-8
19-8	PORDBGMSR Field Descriptions.....	19-10
19-9	PORDEVSR2 Field Descriptions .....	19-11
19-10	GPPORCR Field Descriptions .....	19-11
19-11	GPIOCR Field Descriptions.....	19-12
19-12	GPOUTDR Field Descriptions .....	19-12
19-13	GPINDR Field Descriptions .....	19-13
19-14	PMUXCR Field Descriptions .....	19-14

# Tables

Table Number	Title	Page Number
19-15	DEVDISR Field Descriptions .....	19-14
19-16	POWMGTCSR Field Descriptions .....	19-16
19-17	MCPSUMR Field Descriptions .....	19-18
19-18	RSTRSCR Field Descriptions.....	19-18
19-19	PVR Field Descriptions .....	19-19
19-20	SVR Field Descriptions .....	19-19
19-21	RSTCR Field Descriptions.....	19-20
19-22	LBCVSELCR Field Descriptions .....	19-20
19-23	DDRCR Field Descriptions .....	19-21
19-24	DDRCDR Field Descriptions.....	19-22
19-25	DDRCLKDR Field Descriptions .....	19-22
19-26	CLKOCR Field Descriptions .....	19-23
19-27	SRDS1CR1 Field Descriptions.....	19-24
19-28	SRDS2CR1 Field Descriptions.....	19-25
19-29	MPC8533E Power Management Modes—Basic Description .....	19-26
19-30	Power Management Entry Protocol and Initiating Functional Units .....	19-29
20-1	Control Register Memory Map .....	20-4
20-2	PMGC0 Field Descriptions.....	20-5
20-3	PMLCA0 Field Descriptions .....	20-6
20-4	PMLCA1–PMLCA9 Field Descriptions.....	20-7
20-5	PMLCB0 Field Descriptions.....	20-8
20-6	PMLCB $n$ Field Descriptions.....	20-9
20-7	PMC0 Field Descriptions.....	20-11
20-8	PMC[1–9] Field Descriptions .....	20-11
20-9	Burst Definition.....	20-14
20-10	Performance Monitor Events .....	20-17
20-11	PMGC0 and PMLCAn Settings.....	20-28
20-12	Register Settings for Counting Examples .....	20-28
21-1	POR Configuration Settings and Debug Modes .....	21-3
21-2	Debug, Watchpoint and Test Signal Summary.....	21-5
21-3	Debug Signals—Detailed Signal Descriptions .....	21-6
21-4	Watchpoint and Trigger Signals—Detailed Signal Descriptions .....	21-7
21-5	JTAG Test and Other Signals—Detailed Signal Descriptions.....	21-8
21-6	Debug and Watchpoint Monitor Memory Map.....	21-9
21-7	WMCR0 Field Descriptions.....	21-11
21-8	WMCR1 Field Descriptions.....	21-12
21-9	WMAR Field Descriptions .....	21-13
21-10	WMAMR Field Descriptions.....	21-14
21-11	WMTMR Field Descriptions .....	21-14
21-12	Transaction Types by Interface .....	21-15
21-13	WMSR Field Descriptions .....	21-16

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
21-14	TBCR0 Field Descriptions .....	21-17
21-15	TBCR1 Field Descriptions .....	21-18
21-16	TBAR Field Descriptions .....	21-19
21-17	TBAMR Field Descriptions .....	21-19
21-18	TBTMR Field Descriptions .....	21-20
21-19	TBSR Field Descriptions .....	21-21
21-20	TBACR Field Descriptions .....	21-22
21-21	TBADHR Field Descriptions .....	21-22
21-22	TBADR Field Descriptions .....	21-23
21-23	PCIDR Field Descriptions .....	21-23
21-24	CCIDR Field Descriptions .....	21-24
21-25	TOSR Field Descriptions .....	21-25
21-26	Source and Target ID Values .....	21-26
21-27	CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000) .....	21-29
21-28	DDR Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 001) .....	21-29
21-29	PCI Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 010) .....	21-30
21-30	PCI Express Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 100 or 101 or 110) .....	21-30
B-1	Memory Map .....	B-1



# Tables

**Table  
Number**

**Title**

**Page  
Number**

## About This Book

This reference manual defines the functionality of the MPC8533E. This device integrates an e500v2 core, built on Power Architecture™ technology, with system logic required for networking, telecommunications, and wireless infrastructure applications. The core is a low-power implementation of resources defined by the Power ISA (instruction set architecture) for embedded processors. This book is intended as a companion to the e500 core family reference manual.

## Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

## Organizations

Following is a summary and a brief description of the major parts of this reference manual:

Part I, “Overview,” describes the many features of the MPC8533E integrated host processor at an overview level. The following chapters are included:

- [Chapter 1, “Overview,”](#) provides a high-level description of features and functionality of the device, including its interfaces, and its programming model. The functional operation of the device, with emphasis on peripheral functions is also described.
- [Chapter 2, “Memory Map,”](#) describes the device memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.
- [Chapter 3, “Signal Descriptions,”](#) lists all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).
- [Chapter 4, “Reset, Clocking, and Initialization,”](#) describes the hard and soft resets, the power-on reset (POR) sequence, power-on reset configuration, clocking, and initialization of the MPC8533E.

Part II, “e500 Core Complex and L2 Cache,” describes the many features of the device core at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- [Chapter 5, “Core Complex Overview,”](#) provides an overview of the e500v2 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- [Chapter 6, “Core Register Summary,”](#) lists the e500v2 registers in reference form.

- [Chapter 7, “L2 Look-Aside Cache/SRAM,”](#) describes the L2 cache. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

Part III, “Memory, Security, and I/O Interfaces,” defines the device’s memory, security, and I/O interfaces and how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- [Chapter 8, “e500 Coherency Module,”](#) defines the e500v2 coherency module and how it facilitates communication between the e500v2 core complex, the L2 cache, and the other blocks that comprise the coherent memory domain of the MPC8533E.

The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500v2 core to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500v2- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8533E.

- [Chapter 9, “DDR Memory Controller,”](#) describes the DDR/DDR2 SDRAM memory controller of the MPC8533E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. Special features like ECC error injection support rapid system debug.
- [Chapter 10, “Programmable Interrupt Controller,”](#) describes the embedded programmable interrupt controller (PIC) of the MPC8533E. The PIC is OpenPIC-compliant, provides interrupt management, and receives hardware-generated interrupts from different sources (both internal and external), prioritizing them and delivering them to the CPU for servicing.
- [Chapter 11, “I<sup>2</sup>C Interfaces,”](#) describes the inter-IC (IIC or I<sup>2</sup>C) bus controllers of the MPC8533E. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters and LCDs. The MPC8533E powers up in boot sequencer mode, which allows the I<sup>2</sup>C1 controller to initialize configuration registers.
- [Chapter 12, “Security Engine \(SEC\) 2.1,”](#) describes the security controller of the MPC8533E.
- [Chapter 13, “DUART,”](#) describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- [Chapter 14, “Local Bus Controller,”](#) describes the local bus controller of the MPC8533E. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller controls eight memory banks shared by a high-performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.
- [Chapter 15, “Enhanced Three-Speed Ethernet Controllers,”](#) describes the two enhanced three-speed Ethernet controllers on the MPC8533E. These controllers provide 10/100/1GByte Ethernet support with a complete set of media-independent interface options including MII, RMII, GMII, RGMII, SGMII, TBI, and RTBI. Each controller provides very high throughput using a

captive DMA channel and direct connection to the MPC8533E memory coherency module. The controllers provide two full-duplex FIFO interface modes and quality of service support. They are backward compatible with PowerQUICC™ III TSEC controllers.

- [Chapter 16, “DMA Controller,”](#) describes the four-channel general-purpose DMA controller of the MPC8533E. The DMA controller transfers blocks of data independent of the e500v2 core or external hosts. Data movement occurs among the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- [Chapter 17, “PCI Bus Interface,”](#) describes the PCI controller of the MPC8533E.
- [Chapter 18, “PCI Express Interface Controller,”](#) describes the PCI-Express implementation of the MPC8533E.

Part IV, “Global Functions and Debug,” defines other global blocks of the MPC8533E. The following chapters are included:

- [Chapter 19, “Global Utilities,”](#) defines the global utilities of the MPC8533E. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals.
- [Chapter 20, “Device Performance Monitor,”](#) describes the performance monitor of the MPC8533E. Note that the device performance monitor is similar to but separate from the performance monitor on the e500v2 core.
- [Chapter 21, “Debug Features and Watchpoint Facility,”](#) describes the debug features and watchpoint monitor of the MPC8533E.

This manual contains the following appendixes:

- [Appendix A, “Revision History,”](#) lists the major differences between revisions of the *MPC8533E PowerQUICC III Integrated Host Processor Family Reference Manual*.
- [Appendix B, “Complete List of Configuration, Control, and Status Registers,”](#) contains a complete listing of the memory-mapped registers of the MPC8533E.

This reference manual contains a glossary and a general index.

## Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the architecture.

## General Information

The following documentation, published by Morgan-Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA, provides useful information about the PowerPC™ architecture and computer architecture in general:

- *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, Second Edition, by International Business Machines, Inc.

- *Computer Architecture: A Quantitative Approach*, Third Edition, by John L. Hennessy and David A. Patterson
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, by David A. Patterson and John L. Hennessy

## Related Documentation

Freescale documentation is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:

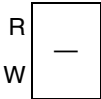
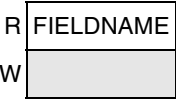
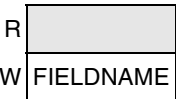
- The e500 core family reference manual—This book describes the functionality of the e500 embedded microprocessor core for software and hardware developers.
- *EREF: A Programmer's Reference Manual for Freescale Book E Processors*—This book describes the programming model for embedded processors as it is defined by the Power ISA and the Freescale Embedded Implementation Standards (this book will be replaced by *EREF: A Programmer's Reference Manual for Freescale Embedded Processors (Including the e200, e500, and e700 Families)*).
- Reference manuals (formerly called user's manuals)—These books provide details about individual implementations.
- Addenda/errata to reference or user's manuals—Because some processors have follow-on parts, an addendum describes the additional features and functionality changes. These addenda are intended for use with the corresponding reference or user's manuals.
- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product Briefs—Each device has a technical summary that provides an overview of its features.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale processors.

Additional literature is published as new processors become available. For a current list of documentation, refer to <http://www.freescale.com>.



# Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
<b>mnemonics</b>	Instruction mnemonics are shown in lowercase bold.
<i>italics</i>	<p>Italics indicate variable command parameters, for example, <b>bcctrx</b>.</p> <p>Book titles in text are set in italics</p> <p>Internal signals are set in lowercase italics, for example, <math>\overline{core\_int}</math></p>
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, rB	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REG[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
<i>x</i>	An italicized <i>x</i> indicates an alphanumeric variable.
<i>n</i>	An italicized <i>n</i> indicates a numeric variable.
¬	NOT logical operator
&	AND logical operator
	OR logical operator
	Concatenation, for example TCR[WP]  TCR[WPEXT]
	Indicates a reserved bit field in a memory-mapped or an e500 register.
	Indicates a read-only bit field in a memory-mapped register.
	Indicates a write-only bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.

# Signal Conventions

- OVERBAR** An overbar indicates that a signal is active-low.
- lowercase\_italics* Lowercase italics is used to indicate internal signals.
- lowercase\_plaintext Lowercase plain text is used to indicate signals that are used for configuration. For more information, see [Section 3.2, “Configuration Signals Sampled at Reset.”](#)

# Acronyms and Abbreviations

[Table i](#) contains acronyms and abbreviations used in this document.

**Table i. Acronyms and Abbreviated Terms**

Term	Meaning
ADB	Allowable disconnect boundary
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CEPT	Conférence des administrations européennes des postes et télécommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
ECM	e500 coherency module
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface

**Table i. Acronyms and Abbreviated Terms (continued)**

<b>Term</b>	<b>Meaning</b>
GMI	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
GUI	Graphical user interface
I <sup>2</sup> C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
IrDA	Infrared Data Association
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least significant byte
lsb	Least significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most significant byte
msb	Most significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OSI	Open systems interconnection
PCI	Peripheral component interconnect bus
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer

**Table i. Acronyms and Abbreviated Terms (continued)**

<b>Term</b>	<b>Meaning</b>
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
POR	Power-on reset
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCC	Serial communication controller
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
TSEC	Three-speed Ethernet controller
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
UTP	Unshielded twisted pair
VA	Virtual address
ZBT	Zero bus turnaround

# Part I

## Overview

Part I describes the many features of the MPC8533E integrated host processor at an overview level. The following chapters are included:

[Chapter 1, “Overview,”](#) provides a high-level description of features and functionality of the MPC8533E integrated host processor. It describes the MPC8533E, its interfaces, and programming model. The functional operation of the MPC8533E, with emphasis on peripheral functions, is also described.

[Chapter 2, “Memory Map,”](#) describes the MPC8533E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of each.

[Chapter 3, “Signal Descriptions,”](#) provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).

[Chapter 4, “Reset, Clocking, and Initialization,”](#) describes the hard and soft resets, power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8533E.



# Chapter 1

## Overview

The MPC8533E integrates an e500v2 core, built on Power Architecture™ technology, with system logic required for networking, telecommunications, and wireless infrastructure applications. The MPC8533E is a member of the PowerQUICC™ III family of devices that combine system-level support for industry-standard interfaces with processors that implement Power Architecture™ technology. This chapter provides a high-level description of features and functionality of the MPC8533E integrated processor.

### 1.1 Introduction

The MPC8533E uses the e500 core and high-speed interconnect technology to balance processor performance with I/O system throughput. The e500 core implements embedded resources defined by the Power ISA and provides unprecedented levels of hardware and software debugging support.

Additionally, the MPC8533E offers a double-precision floating-point auxiliary processing unit (APU), 256 Kbytes of level-2 cache, two integrated 10/100/1Gb enhanced three-speed Ethernet controllers (eTSECs) with TCP/IP acceleration and classification capabilities, a DDR/DDR2 SDRAM memory controller, a 32-bit PCI controller, a programmable interrupt controller, two I<sup>2</sup>C controllers, a four-channel DMA controller, an integrated security engine with XOR acceleration, a general-purpose I/O port, and dual universal asynchronous receiver/transmitters (DUART). For high speed interconnect, the MPC8533E provides a set of multiplexed pins that support up three PCI Express interfaces. The high level of integration in the MPC8533E helps simplify board design and offers significant bandwidth and performance.

The MPC8533E is also available without a security engine, in a configuration known as the MPC8533. All specifications other than those relating to security apply to the MPC8533 exactly as described in this document.

## 1.2 MPC8533E Overview

This section provides a high-level overview of MPC8533E features. Figure 1-1 shows the major functional units within the MPC8533E.

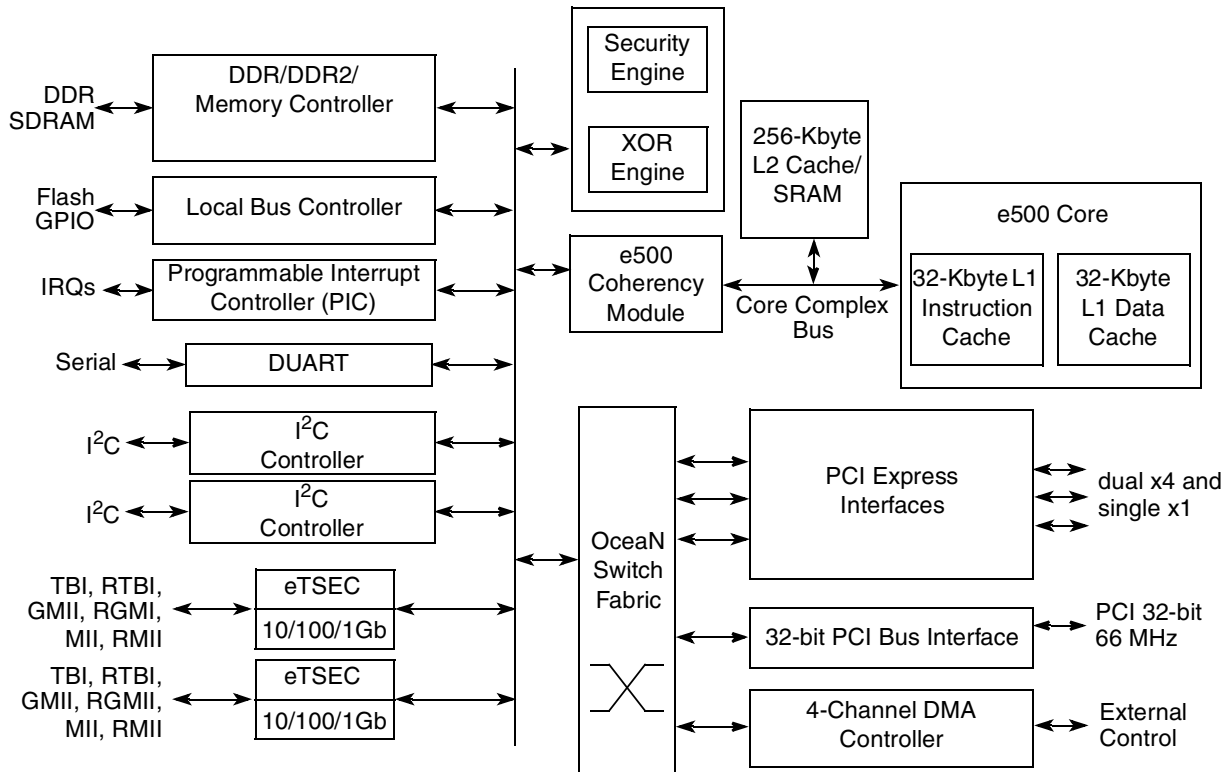


Figure 1-1. MPC8533E Block Diagram

### 1.2.1 Key Features

The following list provides an overview of the MPC8533E feature set:

- High-performance 32-bit e500 core that implements resources for embedded processors defined by the Power ISA:
  - 32-Kbyte L1 instruction cache and 32-Kbyte L1 data cache with parity protection. Caches can be locked entirely or on a per-line basis, with separate locking for instructions and data.
  - Signal-processing engine (SPE) instructions. Extensive instruction set for vector (64-bit) integer and fractional operations. These instructions use both the upper and lower words of the 64-bit general-purpose registers (GPR) as they are defined by the SPE category of the Power ISA.
  - Double-precision (64-bit) floating-point instructions that use the 64-bit GPRs.
  - Embedded vector and scalar single-precision (32-bit) floating-point instructions.
  - 36-bit real addressing (up to 64 Gbytes of memory)
  - Memory management unit (MMU) especially designed for embedded applications that support 4-Kbyte–4-Gbyte page sizes.



The e500 defines features that are not implemented on the MPC8533E. It also generally defines features that the MPC8533E implements more specifically. An understanding of these differences can be critical to ensure proper operations.

Section 1.3.1, “e500 Core Overview,” includes a comprehensive list of e500 core features.

- 256-Kbyte L2 cache/SRAM
  - Flexible configuration. See Section 1.3.2, “On-Chip Memory Unit”
  - Full ECC support on 64-bit boundary in both cache and SRAM modes
  - Cache mode supports instruction caching, data caching, or both.
  - External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing).
    - One, two, or four ways can be configured for stashing only
  - Eight-way set-associative cache organization (32-byte cache lines)
  - Supports locking entire cache or selected lines. Individual line locks are set and cleared through Book E instructions or by externally mastered transactions.
  - Global locking and flash clearing done through writes to L2 configuration registers
  - Instruction and data locks can be flash cleared separately.
  - SRAM features include the following:
    - I/O devices access SRAM regions by marking transactions as snoopable (global).
    - Regions can reside at any aligned location in the memory map.
    - Byte-accessible ECC is protected using read-modify-write transaction accesses for smaller-than-cache-line accesses.
- Address translation and mapping unit (ATMU)
  - Ten local access windows define mapping within local 36-bit address space.
  - Inbound and outbound ATMUs map to larger external address spaces.
    - Three inbound windows plus a configuration window on PCI and PCI Express
    - Four outbound windows plus default translation for PCI and PCI Express
- DDR/DDR2 memory controller
  - Programmable timing supporting DDR and DDR2 SDRAM
  - 64-bit data interface
  - Four banks of memory supported, each up to 4 Gbytes, to a maximum of 16 Gbytes
  - DRAM chip configurations from 64 Mbits to 4 Gbits with x8/x16 data ports
  - Full error checking and correction (ECC) support
  - Page mode support
    - Up to 16 simultaneous open pages for DDR
    - Up to 32 simultaneous open pages for DDR2
  - Contiguous or discontiguous memory mapping
  - Chip-select interleaving support
  - Sleep mode support for self-refresh SDRAM

- On-die termination support when using DDR2
- Supports auto refreshing
- On-the-fly power management using CKE signal
- Registered DIMM support
- Fast memory access via JTAG port
- 2.5-V SSTL\_2 compatible I/O (1.8-V SSTL\_1.8 for DDR2)
- Support for battery-backed main memory
- Programmable interrupt controller (PIC)
  - Programming model is compliant with the OpenPIC architecture.
  - 16 programmable interrupt and processor task priority levels
  - 12 discrete external interrupts
  - 4 message interrupts with 32-bit messages
  - Message shared interrupts
  - Connection of an external interrupt controller such as the 8259 programmable interrupt controller
  - Four global high resolution timers/counters that can generate interrupts
  - Supports a variety of other internal interrupt sources
  - Fully nested interrupt delivery
  - Interrupts can be routed to external pin for external processing.
  - Interrupts can be routed to the e500 core's standard or critical interrupt inputs.
  - Interrupt summary registers allow fast identification of interrupt source.
- Integrated security engine (SEC) optimized to process all the algorithms associated with IPSec, IKE, WTLS/WAP, SSL/TLS, and 3GPP
  - Four crypto-channels, each supporting multi-command descriptor chains
    - Dynamic assignment of crypto-execution units via an integrated controller
    - Buffer size of 256 bytes for each execution unit, with flow control for large data sizes
  - PKEU—public key execution unit
    - RSA and Diffie-Hellman; programmable field size up to 2048 bits
    - Elliptic curve cryptography with  $F_2^m$  and  $F(p)$  modes and programmable field size up to 511 bits
  - DEU—Data Encryption Standard (DES) execution unit
    - DES, 3DES
    - Two key (K1, K2, K1) or three key (K1, K2, K3)
    - ECB and CBC modes for both DES and 3DES
  - AESU—Advanced Encryption Standard (AES) unit
    - Implements the Rijndael symmetric key cipher
    - ECB, CBC, CTR, and CCM modes
    - 128-, 192-, and 256-bit key lengths

- AFEU—ARC four execution unit
  - Implements a stream cipher compatible with the RC4 algorithm
  - 40- to 128-bit programmable key
- MDEU—message digest execution unit
  - SHA with 160- or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- KEU—Kasumi execution unit
  - Implements F8 algorithm for encryption and F9 algorithm for integrity checking
  - Also supports A5/3 and GEA-3 algorithms
- RNG—random number generator
- XOR engine accelerates parity checking in RAID storage applications.
- Dual I<sup>2</sup>C controllers
  - Two-wire interface
  - Multiple master support
  - Master or slave I<sup>2</sup>C mode support
  - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
  - Optionally loads configuration data from serial ROM at reset via the I<sup>2</sup>C1 interface
  - Can be used to initialize configuration registers and/or memory
  - Supports extended I<sup>2</sup>C addressing mode
  - Data integrity checked with preamble signature and CRC
- DUART
  - Two 4-wire interfaces (SIN, SOUT,  $\overline{\text{RTS}}$ ,  $\overline{\text{CTS}}$ )
  - Programming model compatible with the original 16450 UART and the PC16550D
- Local bus controller (LBC)
  - Multiplexed 32-bit address and data bus operating at up to 166 MHz
  - Eight chip selects support eight external slaves.
  - Up to eight-beat burst transfers
  - The 32-, 16-, and 8-bit port sizes are controlled by an on-chip memory controller.
  - Three protocol engines available on a per chip select basis:
    - General-purpose chip select machine (GPCM)
    - Three user programmable machines (UPMs)
    - Dedicated single data rate SDRAM controller
  - Parity support
  - Default boot ROM chip select with configurable bus width (8, 16, or 32 bits)

- Two enhanced three-speed Ethernet controllers (eTSECs)
  - Backward compatible with MPC8540/MPC8560 (PowerQUICC III) TSEC
  - Three-speed support (10/100/1000 Mbps)
  - Two IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant controllers
  - Support for two full-duplex FIFO interface modes
  - Support for various Ethernet physical interfaces:
    - 1000 Mbps full-duplex IEEE 802.3 GMII, IEEE 802.3z TBI, RTBI, and RGMII
    - 10/100 Mbps full and half-duplex IEEE 802.3 MII, IEEE 802.3 RGMII, and RMI
  - Flexible configuration for multiple PHY interface configurations. See [Section 1.3.13, “Enhanced Three-Speed Ethernet Controllers \(eTSECs\),”](#) for more information.
  - TCP/IP acceleration and QoS features available
    - IP v4 and IP v6 header recognition on receive
    - IP v4 header checksum verification and generation
    - TCP and UDP checksum verification and generation
    - Per-packet configurable acceleration
    - Recognition of VLAN, stacked (queue in queue) VLAN, IEEE 802.2 Std.<sup>TM</sup>, PPPoE session, MPLS stacks, and ESP/AH IP-security headers
    - Supported in all FIFO modes
  - Quality of service support:
    - Transmission from up to eight physical queues
    - Reception to up to eight physical queues
  - Full- and half-duplex Ethernet support (1000 Mbps supports only full duplex):
    - IEEE 802.3 Std.<sup>TM</sup> full-duplex flow control (automatic PAUSE frame generation or software-programmed PAUSE frame generation and recognition)
  - Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 Std.<sup>TM</sup> virtual local area network (VLAN) tags and priority
  - VLAN insertion and deletion
    - Per-frame VLAN control word or default VLAN for each eTSEC
    - Extracted VLAN control word passed to software separately
  - Retransmission following a collision
  - CRC generation and verification of inbound/outbound frames
  - Programmable Ethernet preamble insertion and extraction of up to 7 bytes
  - MAC address recognition:
    - Exact match on primary and virtual 48-bit unicast addresses
    - VRRP and HSRP support for seamless router fail-over
    - Up to 16 exact-match MAC addresses supported
    - Broadcast address (accept/reject)
    - Hash table match on up to 512 multicast addresses

- Promiscuous mode
- Buffer descriptors backward compatible with MPC8260 and MPC860T 10/100 Ethernet programming models
- RMON statistics support
- 10-Kbyte internal transmit and 2-Kbyte receive FIFOs
- MII management interface for control and status
- Ability to force allocation of header information and buffer descriptors into L2 cache
- OCeaN switch fabric
  - Full crossbar packet switch
  - Reorders packets from a source based on priorities
  - Reorders packets to bypass blocked packets
  - Implements starvation avoidance algorithms
  - Supports packets with payloads of up to 256 bytes
- Integrated DMA controller
  - Four-channel controller
  - All channels accessible by both the local and remote masters
  - Extended DMA functions (advanced chaining and striding capability)
  - Support for scatter and gather transfers
  - Misaligned transfer capability
  - Interrupt on completed segment, link, list, and error
  - Supports transfers to or from any local memory or I/O port
  - Selectable hardware-enforced coherency (snoop/no snoop)
  - Ability to start and flow control each DMA channel from external 3-pin interface
  - Ability to launch DMA from single write transaction
- PCI controller
  - PCI 2.2 compatible
  - One 32-bit PCI port with support for speeds of up to 66 MHz
  - Host and agent mode support
  - 64-bit dual address cycle (DAC) support
  - Supports PCI-to-memory and memory-to-PCI streaming
  - Memory prefetching of PCI read accesses
  - Supports posting of processor-to-PCI and PCI-to-memory writes
  - PCI 3.3-V compatible
  - Selectable hardware-enforced coherency
- PCI Express interfaces
  - PCI Express 1.0a compatible
  - Supports dual x4 and single x1 links

- Auto-detection of number of connected lanes
- Selectable operation as root complex or endpoint
- Both 32- and 64-bit addressing
- 256-byte maximum payload size
- Virtual channel 0 only
- Traffic class 0 only
- Full 64-bit decode with 32-bit wide windows
- Power management
  - Supports power saving modes: doze, nap, and sleep
  - Employs dynamic power management, which automatically minimizes power consumption of blocks when they are idle
- System performance monitor
  - Supports eight 32-bit counters that count the occurrence of selected events
  - Ability to count up to 512 counter-specific events
  - Supports 64 reference events that can be counted on any of the eight counters
  - Supports duration and quantity threshold counting
  - Burstiness feature permits counting of burst events with a programmable time between bursts.
  - Triggering and chaining capability
  - Ability to generate an interrupt on overflow
- System access port
  - Uses JTAG interface and a TAP controller to access entire system memory map
  - Supports 32-bit accesses to configuration registers
  - Supports cache-line burst accesses to main memory
  - Supports large block (4-Kbyte) uploads and downloads
  - Supports continuous bit streaming of entire block for fast upload and download
- Designed to be compliant with IEEE 1149.1 Std.<sup>TM</sup>, JTAG boundary scan
- 783 FC-PBGA package

## 1.3 MPC8533E Architecture Overview

The following sections describe the major functional units of the MPC8533E.

### 1.3.1 e500 Core Overview

This device uses the e500 microprocessor core complex. The e500 core has an internal PLL that allows independent optimization of the operating frequencies. The core frequencies are derived from either the

primary PCI clock input or an external oscillator. For background information regarding the e500 core refer to the following documents:

- The e500 core family reference manual
- *PowerPC™ e500 Application Binary Interface User's Guide*
- *EREF: A Programmer's Reference Manual for Freescale Book E Processors*—this book will be replaced by *EREF: A Programmer's Reference Manual for Freescale Embedded Processors (Including the e200, e500, and e700 Families)*

#### NOTE

The e500 defines features that are not implemented on this device. It also generally defines some features that this device implements more specifically. An understanding of these differences can be critical to ensure proper operations.

The following is a brief list of some of the key features of the e500 core complex:

- Implements full Book E 32-bit architecture
- Implements additional instructions, registers, and interrupts defined by auxiliary processing units (APUs). The SPE APU provides an extensive instruction set for 64-bit vector integer and fractional operations. The embedded floating-point APUs provide vector single-precision instructions that operate on operands comprised of two 32-bit elements; the single-precision scalar instructions use only the bottom word. The double-precision floating-point APU provides scalar (64-bit) double-precision floating-point instructions that use the 64-bit GPRs.

#### NOTE

The SPE APU and double-precision floating-point APU functionality is implemented in all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE, double-precision floating-point, or embedded floating-point APU instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale offers a `libcfs1_e500` library that uses SPE instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

- L1 cache structure
  - 32-Kbyte, 32-byte line, eight-way set-associative instruction cache
  - 32-Kbyte, 32-byte line, eight-way set-associative data cache
  - 1.5-cycle cache array access, 3-cycle load-to-use latency
  - Pseudo-LRU replacement algorithm
  - Copy-back data cache
- Dual-dispatch superscalar
- Precise exception handling

- Seven-stage pipeline control
- Instruction unit
  - Twelve-entry instruction queue
  - Full hardware detection of interlocks
  - Dispatch of up to two instructions per cycle
  - Dispatch serialization control
  - Register dependency resolution and renaming
- Branch unit (BU)
  - Dynamic branch prediction
  - Two-entry branch instruction queue (BIQ)
  - Executes all branch and CR logical instruction
- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue
  - In-order retirement of up to two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts and mispredicted branches
- Two simple execution units that perform:
  - Single-cycle add and subtract
  - Single-cycle shift and rotate
  - Single-cycle logical operations
  - Supports integer signal processing operations
- Multiple-cycle execution unit (MU)
  - 4-cycle latency for integer and floating-point multiplication (including integer, fractional, and both vector and scalar floating-point multiply instructions)
  - Variable-latency divide: 4, 11, 19, and 35 cycles for all Book E, SPE, and embedded floating-point divide instructions. Note that the MU allows divide instructions to bypass the second two MU pipeline stages, freeing those stages for other MU instructions to execute in parallel.
  - 4-cycle floating-point multiply
  - 4-cycle floating-point add and subtract
- Double-precision floating-point APU
- Signal processing engine APU (SPE APU). The single instruction multiple data (SIMD) capability provided by the 64-bit execution units (MIU, LSU, SIU1) is not a separate execution unit. The hardware that executes 32-bit Book E instructions also executes the lower half of 64-bit SPE APU instructions.
  - Single-cycle integer add and subtract
  - Single-cycle logical operations
  - Single-cycle shift and rotate



- 4-cycle integer pipelined multiplies
- 4-, 11-, 19-, and 35-cycle integer divides
- 4-cycle SIMD pipelined multiply-accumulate (MAC)
- 64-bit accumulator for MAC operations
- Single-precision vector and scalar floating-point APUs
- Load/store unit (LSU)
  - 3-cycle load latency
  - Fully pipelined
  - Nine-entry load queue allows up to nine load misses before stalling
  - Can continue servicing load hits when load queue is full
  - Six-entry store queue allows full pipelining of stores
- Cache coherency
  - Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)
  - High-speed, on-chip local bus with data tagging
  - 36-bit address bus
  - 60x-like address protocol with address pipelining and retry/copyback
  - Two general-purpose read data, one write data bus
  - 128-bit data plus parity/tags (each data bus)
  - Supports out-of-order reads, in-order writes
  - Little to no data bus arbitration logic required for native systems
  - Supports one-level pipelining of addresses with address-retry responses
- Extended exception handling
  - Supports Book E interrupt model
    - Interrupt vector prefix register (IVPR)
    - Interrupt vector offset registers (IVORs) 0–15 as defined in Book E plus e500-defined IVORs 32–35
    - Exception syndrome register (ESR)
    - Book E–defined preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rftci** instruction
  - e500-specific interrupts not defined in Book E architecture
    - SPE APU unavailable exception
    - Floating-point data exception
    - Floating-point round exception
    - Performance monitor
- Memory management unit (MMU)
  - Data L1 MMU
    - Four-entry, fully-associative TLB array for variable-sized pages

- 64-entry, four-way set-associative TLB for 4-Kbyte pages
- Instruction L1 MMU
  - Four-entry, four-way set-associative TLB for 4-Kbyte pages
- Unified L2 MMU
  - 16-entry, fully-associative TLB array for variable-sized pages
  - 512-entry, four-way set-associative TLB for 4-Kbyte pages
- Software reload for TLBs
- Support for as much as 4 Gbytes ( $2^{32}$ ) of virtual memory
- Support for as much as 64 Gbytes ( $2^{36}$ ) of physical memory
- Support for big-endian and true little-endian memory on a per-page basis
- Power management
  - Dynamic power management on the core minimizes power consumption of functional units, such as execution units, caches, and MMUs, when they are idle.
  - Core power-saving modes: core-halted and core-stopped
  - NAP, DOZE, and SLEEP bits in HID0 that can be used to assert *nap*, *doze*, and *sleep* core output signals to initiate power-saving modes at the integrated device level.
- Testability
  - LSSD scan design
  - JTAG interface
  - ESP support

### 1.3.2 On-Chip Memory Unit

The MPC8533E contains an internal 256-Kbyte memory array that can be configured as memory-mapped SRAM or as a look-aside L2 cache. The array can also be divided into two arrays, one of which may be used as cache and the other as SRAM.

The memory controller for this array connects to the core complex bus (CCB) and communicates via 128-bit read and write buses to the e500 core and the MPC8533E system logic.

The on-chip memory unit contains:

- 256 Kbytes of on-chip memory
  - L2 cache partitioning is configurable
    - Can act as a 256-Kbyte L2 cache
      - 256-Kbyte array organized as 1024 eight-way sets of 32-byte cache lines
    - Array can be partitioned into either L2 cache or memory mapped SRAM on a 1-, 2-, 4-, or 8-way basis
    - Stashing of I/O data into the L2 array is supported, but can be limited to a 1-, 2-, or 4-way basis
  - SRAM operation is byte-accessible.
  - Data ECC on 64-bit boundaries (single-error correction, double-error detection)

- Tag parity (1 bit covering all tag bits)
- Cache mode supports instruction caching, data caching, or both.
- External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types.
- Separate locking for instructions and data so that locks can be set and cleared separately
- Supports locking the entire cache or selected lines
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges.
- Flash clearing done through writes to L2 configuration registers
- Locks for the entire cache may be set and cleared by accesses to memory-mapped control registers.

### 1.3.3 On-Chip Memory as Memory-Mapped SRAM

When the on-chip memory is configured as an SRAM, the memory can be configured to reside at any aligned location in the memory map. It is byte-accessible, and fully ECC protected using read-modify-write transactions for sub-cache-line transactions. I/O devices can access the SRAM by marking transactions global so that they are directed to the CCB.

### 1.3.4 On-Chip Memory as L2 Cache

The on-chip memory arrays include a 256-Kbyte data array, an address tag array, and a status array.

The data array is organized as 1024 sets of eight cache lines. Each cache line size is 32 bytes. The replacement policy within each eight-way set is governed by a pseudo-LRU algorithm. The data is protected with ECC and the tag array is protected by parity.

The L2 cache tags are non-blocking for efficient load/store and snooping operations. The L2 cache can be accessed internally while a load miss is pending (allowing hits under misses). Subsequent to a load miss updating the memory, loads or stores can occur to that line on the very next cycle.

The L2 status array maintains status bits for each line that are used to determine the status of the line. Different combinations of these bits result in different L2 states. Note that because the cache is always write-through, there is no modified state. The status bits include:

- V—Valid
- IL—Instruction locked
- DL—Data locked

All accesses to the L2 memory are fully pipelined so back-to-back loads and stores can have single-cycle throughput.

The cache can be configured to allocate instructions only, data only, or both. It can also be configured to allocate global I/O writes that correspond to a programmable address window or that use a special transaction type (stashing). In this way, DMA engines or I/O devices can force data into the cache.

Line locks can be set in a variety of ways. The Book E architecture defines instructions that explicitly set and clear locks in the L2. These instructions are supported by the core complex and the L2 controller. In addition, the L2 controller can be configured to lock all lines that fall into either of two specified address ranges when the line is allocated. Finally, the entire cache can be locked by writing to a configuration register in the L2 cache controller.

The status array tracks line locks as either instruction locks or data locks for each line, and the status array supports flash clearing of all instruction locks or data locks separately by writes to configuration registers in the L2 controller.

### 1.3.5 e500 Coherency Module (ECM)

The e500 coherency module (ECM) provides a mechanism for I/O-initiated transactions to snoop the bus between the e500 core and the integrated L2 cache in order to maintain coherency across local cacheable memory. It also provides a flexible switch-type structure for core- and I/O-initiated transactions to be routed or dispatched to target modules on the device.

### 1.3.6 DDR SDRAM Controller

The MPC8533E supports DDR and DDR2 SDRAM. The memory interface controls main memory accesses and provides for a maximum of 16 Gbytes of main memory. The memory controller can be configured to support the various memory sizes through software initialization of on-chip configuration registers.

The MPC8533E supports a variety of SDRAM configurations. SDRAM banks can be built using DIMMs or directly-attached memory devices. Sixteen multiplexed address signals provide for device densities of 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbits, 2 Gbits and 4 Gbits. Four chip select signals support up to four banks of memory. The MPC8533E supports bank sizes from 64 Mbytes to 4 Gbytes. Nine column address strobes (MDM[0:8]) are used to provide byte selection for memory bank writes.

The MPC8533E can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 16 simultaneously open pages (32 for DDR2) can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save 3 to 4 clock cycles from subsequent burst accesses that hit in an active page.

The MPC8533E supports error checking and correction (ECC) for system memory. Using ECC, the MPC8533E detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

The MPC8533E can invoke a level of system power management by asserting the MCKE SDRAM signal on-the-fly to put the memory into a low-power sleep mode.

The MPC8533E offers the following options to support battery-backed main memory:

- Hardware based

An external voltage sense device is connected to the MPC8533E via an interrupt line. The external interrupt from this device is steered through the MPC8533E interrupt controller to the IRQ\_OUT signal. The IRQ\_OUT signal from the interrupt controller is steered to an enable bit in the DDR controller which immediately causes main memory to enter self-refresh mode.

This proposal precludes any other simultaneous use of IRQ\_OUT.

- Software based

The DDR controller also has a software-programmable bit that immediately puts main memory into self-refresh mode.

It is expected that a critical interrupt routine triggered by an external voltage sense device will have time to set this bit.

The DDR controller offers an initialization bypass feature which system designers may use to prevent re-initialization of main memory during system power-on following abnormal shutdown.

### 1.3.7 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) implements the necessary functions to provide a flexible solution for a general-purpose interrupt control. The interrupt controller unit implements the logic and programming structures of the OpenPIC architecture. The MPC8533E interrupt controller unit supports its processor core and provides for 12 external interrupts (with fully nested interrupt delivery), four message interrupts, internal-logic driven interrupts, and four global high resolution timers. Up to 16 programmable interrupt priority levels are supported.

The MPC8533E supports reception of interrupt messages from the PCI Express interfaces. The PIC supports INTx and message shared interrupts (MSI) from the PCI Express interfaces. Four INTx interrupts are presented as independent PIC interrupt sources. There are 256 individual MSI interrupt sources supported as eight groups of 32 sources. Each of the eight groups constitutes an independent PIC interrupt source. Individual interrupt sources have associated with them a vector, priority and destination value.

The interrupt controller unit can be bypassed to allow use of an external interrupt controller. Inter-processor interrupt (IPI) communication is supported through the external interrupt and core reset signals of different processor cores on the same device. The four IPIs are only used for self-interrupt in a single-core device such as the MPC8533E.

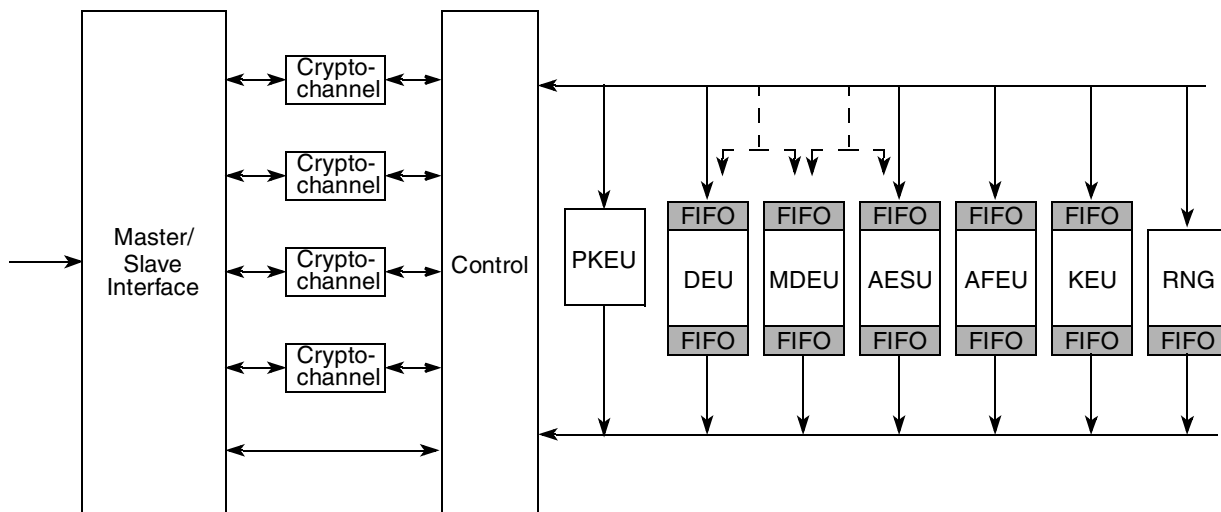
### 1.3.8 Integrated Security Engine (SEC) for the MPC8533E

#### NOTE

The features described in this section are only available on the MPC8533E.  
The MPC8533 does not include an integrated security engine.

The SEC is a modular and scalable security core optimized to process all the algorithms associated with IPsec, IKE, WTLS/WAP, SSL/TLS, and 3GPP. Although it is not a protocol processor, the SEC is designed to perform multi-algorithmic operations (for example, 3DES-HMAC-SHA-1) in a single pass of the data, and every effort has been made to provide the SEC with the flexibility to perform single-pass operations for current and emerging security protocols. The version of the SEC used in the MPC8533E is specifically capable of performing single-pass security cryptographic processing for SSL 3.0, SSL 3.1/TLS 1.0, IPsec, SRTP, and IEEE Std. 802.11i™.

A block diagram of the integrated security engine's internal architecture is shown in Figure 1-2. The bus interface module is designed to transfer 64-bit words between the internal bus and any register inside the SEC.



**Figure 1-2. Integrated Security Engine Functional Blocks**

An operation begins with a write of a pointer to a crypto-channel fetch register that points to a data packet descriptor. The channel requests the descriptor and decodes the operation to be performed. The channel then requests the controller to assign crypto-execution units and fetch the keys, IVs, and data needed to perform the given operation. The controller satisfies the requests by assigning execution units to the channel and by making requests to the master interface. As data is processed, it is written to the individual execution unit's output buffer and then back to system memory via the bus interface module.

The SEC functionality is compatible with code written for the integrated security engine present in the Freescale MPC8541E and MPC8555E devices.

### 1.3.9 I<sup>2</sup>C Controllers

The MPC8533E provides two inter-IC (IIC or I<sup>2</sup>C) interfaces. The I<sup>2</sup>C bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I<sup>2</sup>C allows the MPC8533E to exchange data with other I<sup>2</sup>C devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus (serial data SDA and serial clock SCL) minimizes the interconnections between devices. I<sup>2</sup>C allows the connection of additional devices to the bus for expansion and system development.

The I<sup>2</sup>C controller is a true multimaster bus which includes collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. The I<sup>2</sup>C controller consists of a transmitter/receiver unit, a clocking unit, and a control unit. The dual I<sup>2</sup>C units support general broadcast mode, and on-chip filtering rejects spikes on the bus.

### 1.3.10 Boot Sequencer

The MPC8533E provides a boot sequencer that uses the I<sup>2</sup>C interface to access an external serial ROM and loads the data into the MPC8533E configuration registers. The boot sequencer is enabled by a configuration pin sampled at the negation of the MPC8533E hardware reset signal. If enabled, the boot sequencer holds the MPC8533E processor core in reset until the boot sequence is complete. If the boot sequencer is not enabled, the processor core exits reset and fetches boot code in default configurations.

### 1.3.11 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8533E includes a DUART intended for use in maintenance, bringing-up, and debugging of systems. The MPC8533E provides a standard four-wire handshake (SIN, SOUT,  $\overline{\text{RTS}}$ ,  $\overline{\text{CTS}}$ ) for each port. The DUART is a slave interface. An interrupt is provided to the interrupt controller or optionally steered externally to allow device handshakes. Interrupts are generated for transmit, receive, line status, and MODEM status.

The MPC8533E DUART supports full-duplex operation. It is compatible with the PC16450 and PC16550 programming models. Also, 16-byte FIFOs are supported for both the transmitter and the receiver.

Software programmable baud generators divide the system clock to generate a 16x clock. Serial interface data formats (data length, parity, 1/1.5/2 stop bits, baud rate) are also software selectable.

### 1.3.12 Local Bus Controller

The MPC8533E local bus controller (LBC) port allows connections with a wide variety of external memories, DSPs, and ASICs. Three separate state machines share the same external pins and can be programmed separately to access different types of devices. The general-purpose chip select machine (GPCM) controls accesses to asynchronous devices using a simple handshake protocol. The user programmable machine (UPM) can be programmed to interface to synchronous devices or custom ASIC interfaces. The SDRAM controller provides access to standard SDRAM. Each chip select can be configured so that the associated chip interface can be controlled by the GPCM, UPM, or SDRAM controller. All may exist in the same system.

The GPCM provides a flexible asynchronous interface to SRAM, EPROM, FEPRM, ROM, and other devices such as asynchronous DSP host interfaces and CAMs. Minimal glue logic is required. Handshake signals can be configured to transition on fractions of the system clock. The GPCM does not support bursting.

The UPM allows an extremely flexible interface in which the programmer configures each of a set of general-purpose protocol signals by writing the transition pattern into a memory array. The UPM supports synchronous and bursting interfaces. It also supports multiplexed addressing so that a simple DRAM interface can be implemented. The UPM is entirely flexible in order to provide a very high degree of customization with respect to both asynchronous and burst-synchronous interfaces, which permits glueless or almost glueless connection to burst SRAM, custom ASIC, and synchronous DSP interfaces.

The LBC provides a synchronous DRAM (SDRAM) machine that provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices. An internal PLL (phase-locked loop) for bus clock generation ensures improved data setup margins for board designs. The SDRAM

machine can optimize burst transfers and exploits interleaving to maximize data transfer bandwidth and minimize access latency. Programmable row and column address multiplexing allows a variety of SDRAM configurations and sizes to be supported without hardware changes.

### 1.3.13 Enhanced Three-Speed Ethernet Controllers (eTSECs)

The MPC8533E has two on-chip enhanced three-speed Ethernet controllers. The eTSECs incorporate a media access control (MAC) sublayer that supports 10- and 100-Mbps and 1-Gbps Ethernet/802.3 networks with MII, RMII, GMII, RGMII, TBI, and RTBI physical interfaces. The eTSECs include 2-Kbyte receive and 10-Kbyte transmit FIFOs and DMA functions.

The buffer descriptors are based on the MPC8260 and MPC860T 10/100 Ethernet programming models. Each eTSEC can emulate a PowerQUICC III TSEC, allowing existing driver software to be re-used with minimal change.

The MPC8533E eTSECs support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. Frame headers and buffer descriptors can be forced into the L2 cache to speed classification or other frame processing.

Each eTSEC provides hardware support for accelerating TCP/IP packet transmission and reception. By default, TCP/IP acceleration is not enabled, and the eTSEC processes frames as pure Ethernet frames.

TCP/IP acceleration can be performed at a number of levels. The eTSEC can parse frames at layer 2 of the stack only (Ethernet headers and switching headers), layers 2 to 3 (including IP v4 or IP v6), or layers 2 to 4 (including TCP and UDP).

On receive, the eTSEC provides protocol header recognition, header verification (IP v4 header checksum verification), and TCP/UDP payload checksum verification including verification of associated pseudo-header checksums. On transmit, the eTSEC provides IP v4 and TCP/UDP header checksum generation. The eTSEC does not checksum transmitted packets with IP header options or IP fragments.

To provide for quality of service, transmission from up to eight queues is supported with priority-based queue selection. Arbitration is a modified weighted round-robin queue selection with fair bandwidth allocation.

On receive, packets may be distributed to any of the 64 virtual receive queues overlaid onto the 8 physical receive queues. A table-oriented queue filing strategy is provided based on 16 header fields or flags. Frame rejection is supported for filtering applications.

Filing can be based on Ethernet, IP, and TCP/UDP properties, including VLAN fields, Ether-type, IP protocol type, IP TOS or differentiated services, IP source and destination addresses, TCP/UDP port numbers, or user-defined bit fields.

Each eTSEC provides a full-duplex packet FIFO interface port that bypasses the Ethernet MAC, but re-uses the PHY interface pins. As a result, the FIFO interface normally does not impose the overheads of Ethernet framing. The FIFO interface operates synchronously, at up to 200MHz, providing up to 3.2-Gbps full-duplex transfer rates. Bare IP packets, with an optional 32-bit CRC check sequence, can be transferred to the eTSEC directly. The eTSEC Tx and Rx FIFOs, TCP/IP acceleration functions, and DMA continue to be used in packet FIFO mode.



There are no mode configuration dependencies between eTSEC1 and eTSEC3.

### 1.3.14 Ocean Switch Fabric

In order to reduce the strain on core interconnects with the addition of new functional blocks in this generation of the PowerQUICC family, a multi-port, on-chip, non-blocking crossbar switch fabric called OCeaN (on-chip network) has been provided. The switch fabric serves to decrease contention and increase bandwidth. This non-blocking crossbar fabric allows full-duplex port communication with independent per-port transaction queuing and flow control.

### 1.3.15 Integrated DMA

The MPC8533E DMA engine is capable of transferring blocks of data from any legal address range to any other legal address range. Therefore, it can perform a DMA transfer between any of its I/O or memory ports or even between two devices or locations on the same port.

The four-channel DMA controller allows chaining (both extended and direct) through local memory-mapped chain descriptors. Scattering, gathering, and misaligned transfers are supported. In addition, advanced capabilities such as stride transfers and complex transaction chaining are supported.

DMA transfers can be initiated by a single write to a configuration register. There is also support for external control of transfers using  $\overline{\text{DMA\_DREQ}}$ ,  $\overline{\text{DMA\_DACK}}$ , and  $\overline{\text{DMA\_DDONE}}$  handshake signals.

Interrupts are provided on a completed segment, link, list, chain, or on an error condition. Coherency is selectable and hardware enforced (snoop/no snoop).

### 1.3.16 PCI Controller

The MPC8533E has a 32-bit PCI controller that is compatible with the *PCI Local Bus Specification, Revision 2.2*. The controller can function as a host bridge or as an agent interface.

As a master, the MPC8533E supports read and write operations to PCI memory space, PCI I/O space, and PCI configuration space. The MPC8533E can also generate PCI special-cycle and interrupt-acknowledge commands. As an agent/target, the MPC8533E supports read and write operations to system memory as well as configuration accesses.

An internal arbiter can be used to support up to five external masters. A round robin arbitration algorithm with two priority levels is used.

### 1.3.17 PCI Express Interface

The MPC8533E supports a PCI Express interface compliant with the *PCI Express Base Specification Revision 1.0a*. It is configurable at boot time to act as either root complex or endpoint.

The interface is selectable at boot time to support either 32 or 64-bit addressing. The maximum supported packet payload size is 256 bytes.

The physical layer supports dual x4 links and a single x1 link.

The interface supports virtual channel 0 (VC0) and traffic class 0 (TC0) only.

Inbound INTx transactions are supported and change the state of one of four level-sensitive interrupts presented to the PIC. Outbound INTx transactions are not supported.

Message signaled interrupt (MSI) transactions are supported and control up to 256 interrupt sources within the PIC. Inbound transactions cause specific edge-triggered interrupt sources to be controlled within the PIC. Outbound MSI transactions are created by software using the MSI Capability Register Sets.

The physical layer of the PCI Express interface operates at a 2.5-Gbaud data rate per lane. The theoretical unidirectional peak bandwidth is 16 Gbps. Receive and transmit ports operate independently, resulting in an aggregate theoretical bandwidth of 32 Gbps.

### 1.3.18 Power Management

In addition to low-voltage operation and dynamic power management in its execution units, the MPC8533E supports four power consumption modes: full on, doze, nap, and sleep. The three low-power modes, doze, nap, and sleep, can be entered under software control in the e500 core or by external masters accessing a configuration register.

Doze mode suspends execution of instructions in the e500 core. The core is left in a standby mode in which cache snooping and time base interrupts are still enabled. Device logic external to the processor core is fully functional in this mode.

Nap mode shuts down clocks to all the e500 functional units except the time base, which can be disabled separately. No snooping is performed in nap mode, but the device logic external to the processor core is fully functional.

Sleep mode shuts down not only the e500 core, but also all of the MPC8533E I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the device can be awakened.

### 1.3.19 Clocking

The MPC8533E takes in the SYSCLK signal as an input to the platform PLL and multiplies it to generate the platform clock, which operates at the same frequency as the SDRAM data rate (for example, 666 MHz). The L2 cache also operates at this frequency. The e500 core uses the platform clock as an input to its PLL, which multiplies it again to generate the core clock.

Although the DDR2 controller clocking is source synchronous, a PLL is used in the local bus memory controller (LBC) to generate memory clocks. Six differential clock pairs are generated for DDR SDRAMs. Two clock outputs are generated for the local bus controller (LBC).

### 1.3.20 Address Map

The MPC8533E supports a flexible 36-bit physical address map. Conceptually, the address map consists of local space and external address space. The local address map is 64 Gbytes. The MPC8533E can be made part of a larger system address space through the mapping of translation windows. This functionality is included in the address translation and mapping units (ATMUs). Both inbound and outbound translation

windows are provided. The ATMUs allows the MPC8533E to be part of larger address maps such as the PCI or PCI Express 64-bit address environment.

### 1.3.21 Processing Across the On-Chip Fabric

When processing across the on-chip fabric, the ATMUs at each fabric port are used to determine the flow of data across the MPC8533E. The ATMUs at each fabric port are responsible for generating a fabric port destination ID as well as a new local device address. The port ID and local address are based on the programmed destination of the transaction. The following is a general overview of how the ATMUs process transactions over the on-chip fabric (refer to [Figure 1-3](#)).

1. When a transaction on one of the fabric ports begins, the ATMU on the origination port translates the programmed destination address into both a destination fabric port ID and a local device address.
2. The data is then processed across the on-chip fabric from the origination port to the destination port.
3. If the destination port connects off chip (for example, to a PCI device), the local device address is translated by the destination port ATMU to an outbound address with respect to the destination port's memory map, and the data is processed accordingly.

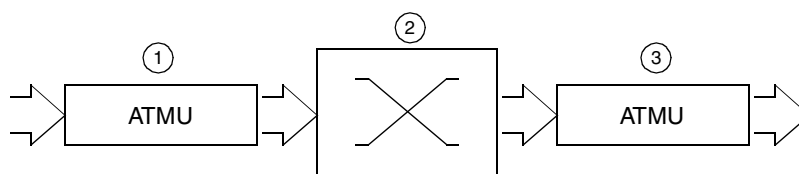


Figure 1-3. Processing Transactions Across the On-Chip Fabric

### 1.3.22 Data Processing with the e500 Coherency Module

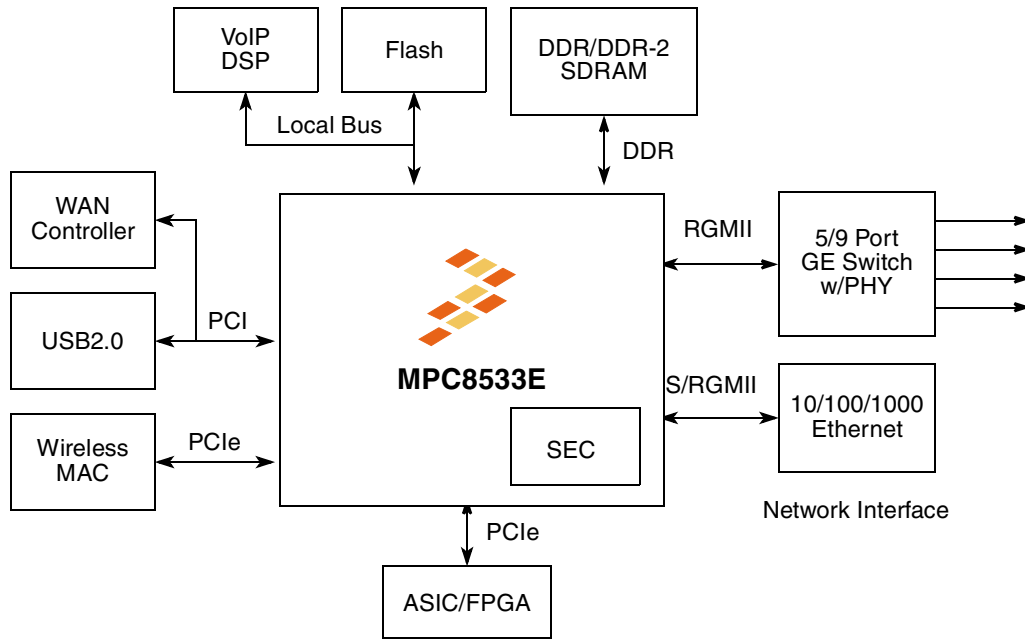
Processing via the ECM is similar to processing across the on-chip fabric (in the sense of how data is received and transmitted) with the exception that the transaction passes through the ECM. The purpose of the ECM is to provide a means for any I/O transaction to maintain coherency with the cacheable DDR SDRAM and the local bus memory. However, simply because the ECM is used does not make transactions across it coherent. The e500 core and L2 cache are snooped to maintain coherency only if the transaction across the ECM is designated as global (GBL bit set). Otherwise, the transaction passes through the ECM using the ECM as a simple conduit to get to its destination. In essence, only global transactions across the ECM are coherent transactions; all others (across the on-chip fabric) are non-coherent.

## 1.4 Application Examples

The MPC8533E is a very flexible device and can be configured to meet many system application needs. The following section provides block diagrams of various applications.

### 1.4.1 Multifunction Router Application

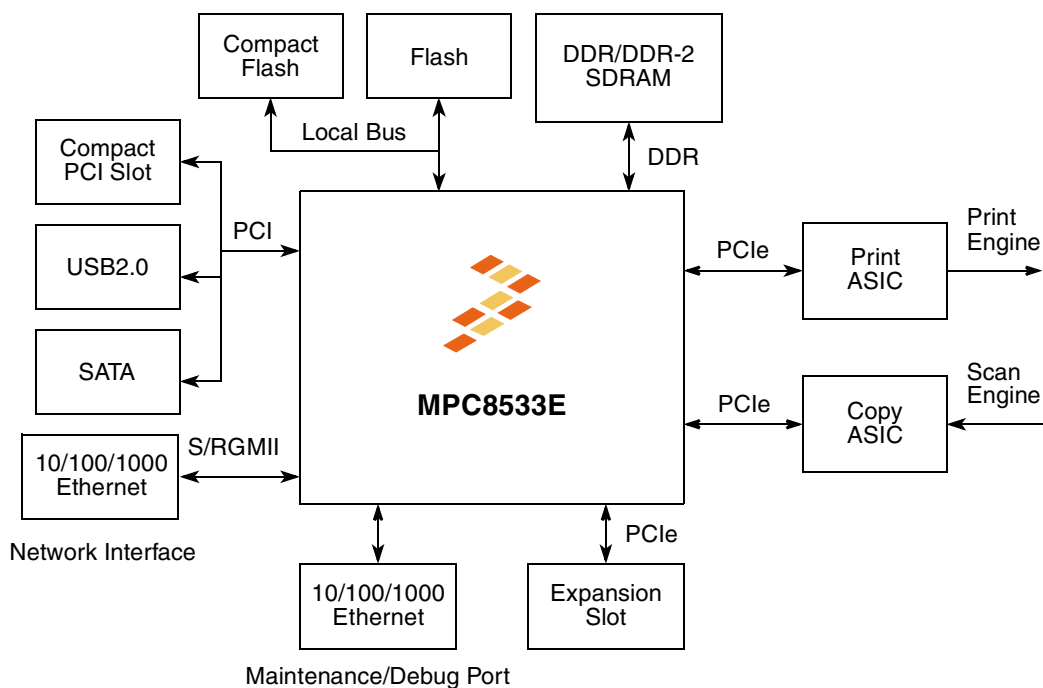
Figure 1-4 shows an MPC8533E in a multi-function router application enabled by local bus, PCI Express, PCI and Ethernet.



**Figure 1-4. Multifunction Router Application Enabled by local bus, PCI Express, PCI, and Ethernet**

## 1.4.2 Multifunction Printer Application

Figure 1-5 illustrates an MPC8533E in a multifunction printer application enabled by local bus, PCI Express, PCI, and Ethernet.



**Figure 1-5. Multifunction Printer Application Enabled by Local Bus, PCI Express, PCI, and Ethernet**

### 1.4.3 Security Appliance

Figure 1-6 shows an MPC8533E as a small-medium enterprise firewall appliance enabled by SEC, local bus, PCI and Ethernet.

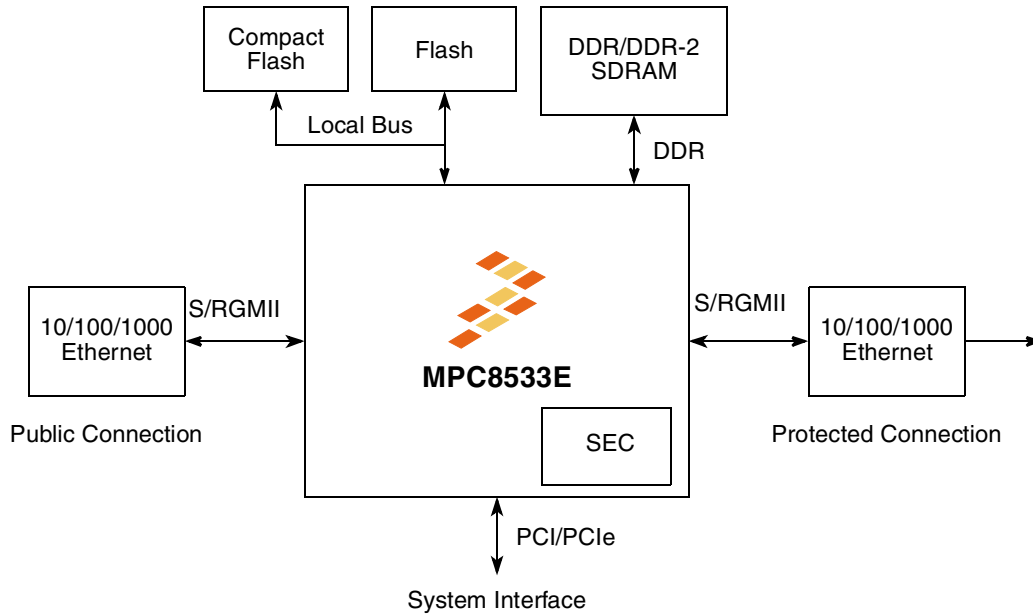


Figure 1-6. Security Appliance Enabled by SEC, local bus, PCI, and Ethernet

## 1.4.4 IP SAN Host Adapter

Figure 1-7 shows the MPC8533E as an IP SAN host adapter enabled by local bus, PCI Express, and Ethernet.

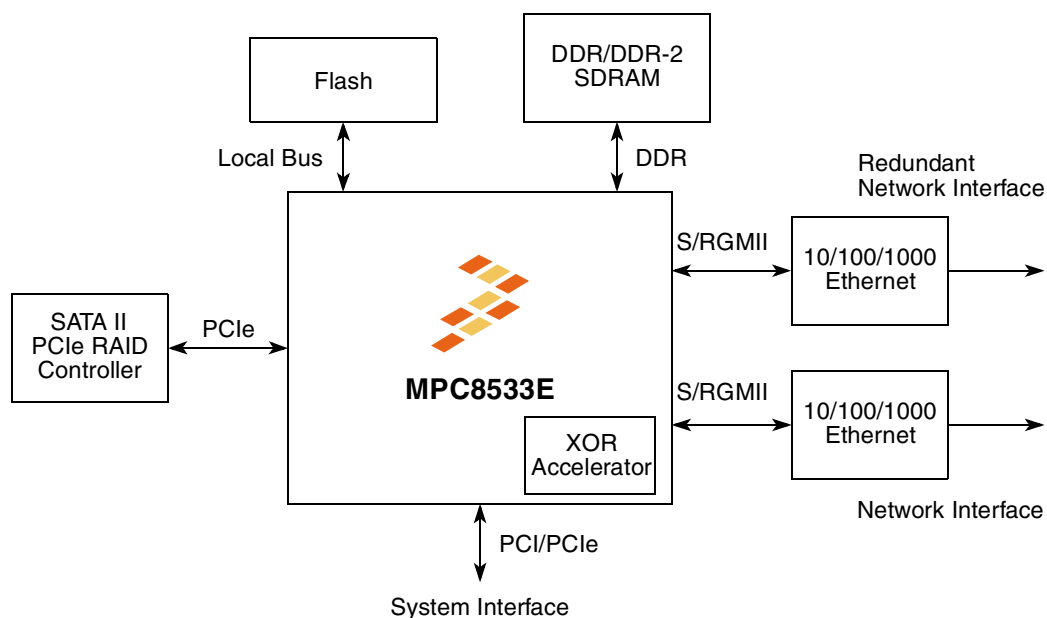


Figure 1-7. IP SAN Host Adapter enabled by local bus, PCI Express, and Ethernet

### 1.4.5 VoIP Aggregation Application

Figure 1-8 shows MPC8533E in a VoIP aggregation application enabled by local bus and Ethernet.

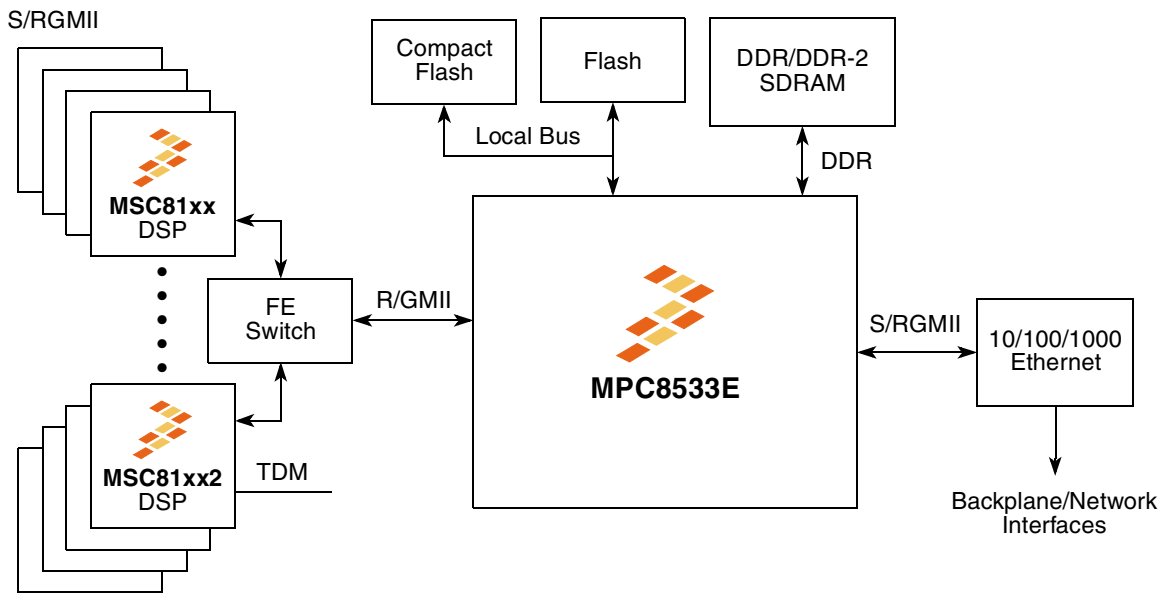


Figure 1-8. VoIP Aggregation Application Enabled by local bus and Ethernet



## Chapter 2 Memory Map

This chapter describes the MPC8533E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.

### 2.1 Local Memory Map Overview and Example

The MPC8533E provides an extremely flexible local memory map. The local memory map refers to the 36-bit address space seen by the processor as it accesses memory and I/O space. DMA engines also see this same local memory map. All memory accessed by the MPC8533E DDR SDRAM and local bus memory controllers exists in this memory map, as do all memory-mapped configuration, control, and status registers.

The local memory map is defined by a set of ten local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. Note that the local access windows do not perform any address translation. The size of each window can be configured from 4 Kbytes to 32 Gbytes. The target interface is specified using the codes shown in [Table 2-1](#).

**Table 2-1. Target Interface Codes<sup>1</sup>**

Target Interface	Target Code
PCI	00000
PCI Express 2	00001
PCI Express 1	00010
PCI Express 3	00011
Local bus	00100
DDR SDRAM	01111

<sup>1</sup> The general intent of the Target Interface Codes is to maintain consistency across PowerQUICC III family devices.

Figure 2-1 shows an example memory map.

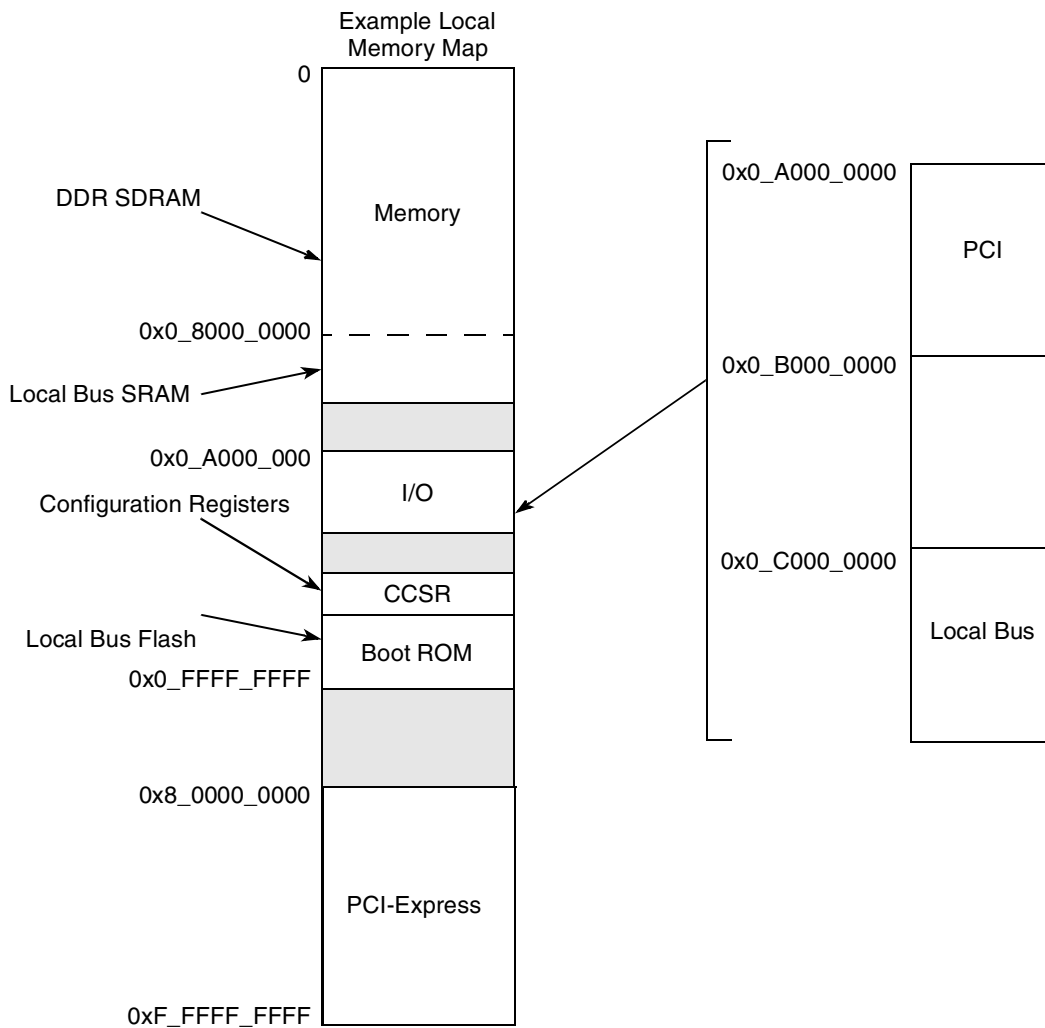


Figure 2-1. Local Memory Map Example

Table 2-2 shows one corresponding set of local access window settings.

Table 2-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
0	0x0_0000_0000	2 Gbytes	0b1111 (DDR SDRAM)
1	0x0_8000_0000	1 Mbyte	0b0100 (local bus)
2	0x0_A000_0000	256 Mbytes	0b0000 (PCI)
3	0x0_C000_0000	256 Mbytes	0b0100 (local bus)
4	0x8_0000_0000	32 Gbytes	0b0100 (PCI Express)
5-9	Unused		

In this example, it is not necessary to use a local access window to specify the range of memory used for memory-mapped registers because this is a fixed 1-Mbyte space pointed to by CCSRBAR. See [Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register \(CCSRBAR\).”](#) Neither is it required to define a local access window to describe the location of the boot ROM because it is in the default location (see [Section 4.4.3.4, “Boot ROM Location”](#)). However, note that the e500 core only provides one default TLB entry to access boot code and it allows for accesses within the highest 4 Kbytes of the low 4 Gbytes of memory. In order for the e500 to access the full 8 Mbytes of default boot space (and the 1 Mbyte of CCSR space), additional TLB entries must be set up within the e500 MMU for mapping these regions.

## 2.2 Address Translation and Mapping

Four distinct types of translation and mapping operations are performed on transactions in the MPC8533E. These are as follows:

- Mapping a local address to a target interface
- Assigning attributes to transactions
- Translating the local 36-bit address to an external address space
- Translating external addresses to the local 36-bit address space

The local access windows perform target mapping for transactions within the local address space. No address translation is performed by the local access windows.

Outbound ATMU windows perform the mapping from the local 36-bit address space to the address spaces of PCI or PCI-Express, for example, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

Inbound ATMU windows perform the address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a similar function as the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the local access windows. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a local access window must also map that same local address to the same target.

All of the configuration registers that define translation and mapping functions use the concept of translation or mapping windows, and all follow the same register format. [Table 2-3](#) summarizes the general format of these window definitions.

**Table 2-3. Format of ATMU Window Definitions**

Register	Function
Translation address	High-order address bits defining location of the window in the target address space
Base address	High-order address bits defining location of the window in the initial address space
Window size/attributes	Window enable, window size, target interface, and transaction attributes

Windows must be a power-of-two size. To perform a translation or mapping function, the address of the transaction is compared with the base address register of each window. The number of bits used in the comparison is dictated by each window's size attribute. When an address hits a window, if address translation is being performed, the new translated address is created by concatenating the window offset to the translation address. Again, the window's size attribute dictates how many bits are translated.

## 2.2.1 SRAM Windows

The on-chip memory array of the MPC8533E can be configured as a memory-mapped SRAM ranging from 64 Kbytes to 256 Kbytes. Configuration registers in the L2 cache controller set the base addresses and sizes for these windows. When enabled, these windows supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR. It is possible to have SRAM windows overlap local access windows, but this is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if the SRAM window overlaps a local access window.

See [Section 7.3.1.3.1, “L2 Memory-Mapped SRAM Base Address Registers 0–1 \(L2SRBARn\),”](#) for information about configuring SRAM windows.

## 2.2.2 Window into Configuration Space

CCSRBAR defines a window used to access all memory-mapped configuration, control, and status registers. No address translation is done, so there are no associated translation address registers. The window is always enabled with a fixed size of 1 Mbyte; no other attributes are attached, so there is no associated size/attribute register. This window always takes precedence over all local access windows. See [Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register \(CCSRBAR\),”](#) and [Section 2.3, “Configuration, Control, and Status Register Map.”](#)

## 2.2.3 Local Access Windows

As demonstrated in the address map overview in [Section 2.1, “Local Memory Map Overview and Example,”](#) local access windows associate a range of the local 36-bit address space with a particular target interface. This allows the internal interconnections of the MPC8533E to route a transaction from its source to the proper target. No address translation is performed. The base address defines the high order address bits that give the location of the window in the local address space. The window attributes enable the window, define its size, and specify the target interface.

With the exception of configuration space (mapped by CCSRBAR), on-chip SRAM regions (mapped by L2SRBAR registers), and default boot ROM, all addresses used by the system must be mapped by a local access window. This includes addresses that are mapped by inbound ATMU windows; target mappings of inbound ATMU windows and local access windows must be consistent.

The local access window registers exist as part of the local access block in the general utilities registers. See [Section 2.3.4, “General Utilities Registers.”](#) A detailed description of the local access window

registers is given in the following sections. Note that the minimum size of a window is 4 Kbytes, so the low order 12 bits of the base address cannot be specified.

### 2.2.3.1 Local Access Register Memory Map

Table 2-4 shows the memory map for the local access registers. In this table and in the register figures and field descriptions, the following access definitions apply:

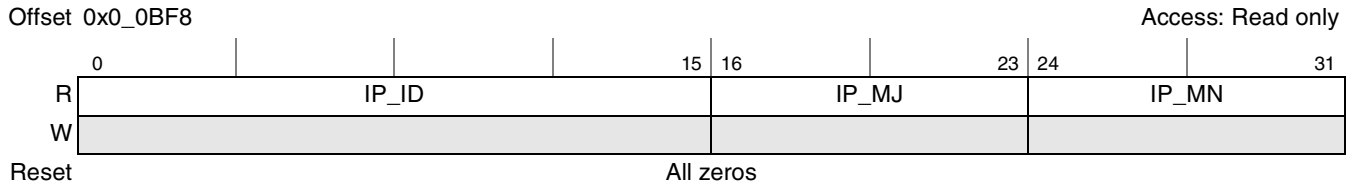
- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 2-4. Local Access Register Memory Map**

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0BF8	LAIPBRR1—Local access IP block revision register 1	R	0x0000_0000	<a href="#">2.2.3.2/2-6</a>
0x0_0BFC	LAIPBRR2—Local access IP block revision register 2	R	0x0000_0000	<a href="#">2.2.3.3/2-6</a>
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0D08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0D10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0D28	LAWBAR9—Local access window 9 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0D30	LAWAR9—Local access window 9 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>

### 2.2.3.2 Local Access IP Block Revision Register 1 (LAIPBRR1)

The local access IP block revision register 1 is shown in [Figure 2-2](#).



**Figure 2-2. Local Access IP Block Revision Register 1 (LAIPBRR1)**

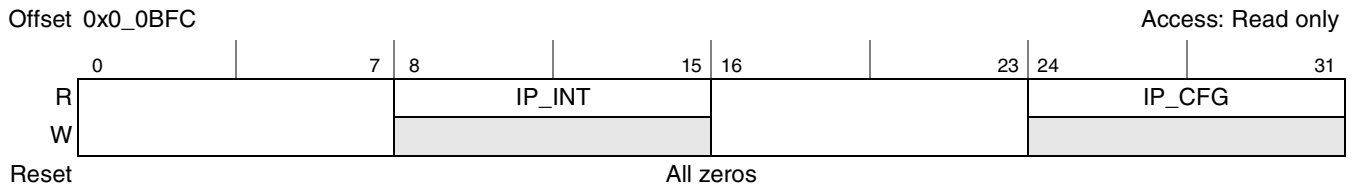
[Table 2-5](#) describes LAIPBRR1 fields.

**Table 2-5. LAIPBRR1 Field Descriptions**

Bits	Name	Description
0–15	IP_ID	IP block ID
16–23	IP_MJ	Major revision
24–31	IP_MN	Minor revision

### 2.2.3.3 Local Access IP Block Revision Register 2 (LAIPBRR2)

The local access IP block revision register 2 is shown in [Figure 2-3](#).



**Figure 2-3. Local Access IP Block Revision Register 2 (LAIPBRR2)**

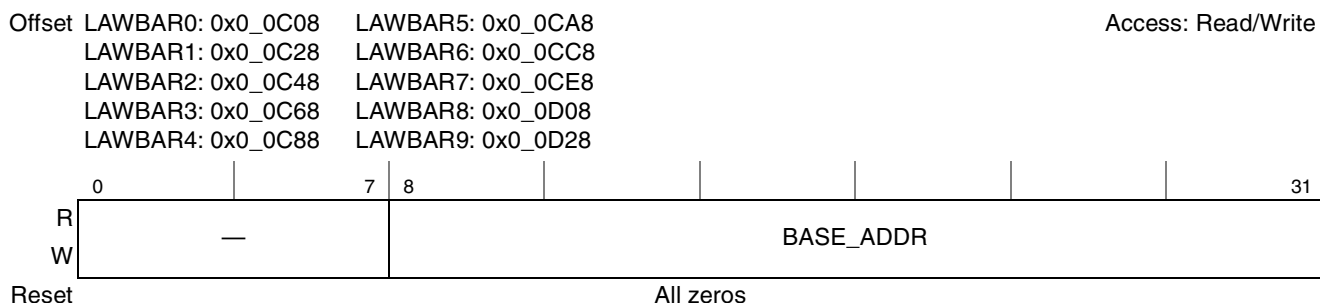
[Table 2-6](#) describes LAIPBRR2 fields.

**Table 2-6. LAIPBRR2 Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

## 2.2.3.4 Local Access Window $n$ Base Address Registers (LAWBAR0–LAWBAR9)

Figure 2-4 shows the bit fields of the LAWBAR $n$  registers.



**Figure 2-4. Local Access Window  $n$  Base Address Registers (LAWBAR0–LAWBAR7)**

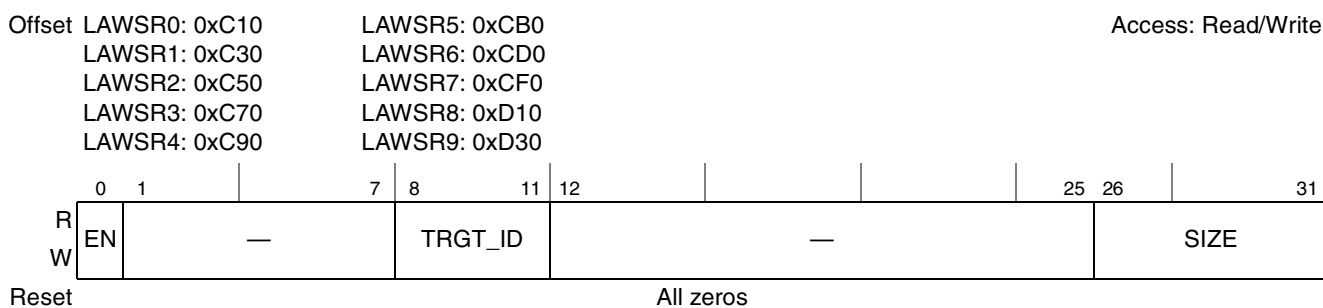
Table 2-7 describes LAWBAR $n$  fields.

**Table 2-7. LAWBAR $n$  Field Descriptions**

Bits	Name	Description
0–7	—	Write reserved, read = 0
8–31	BASE_ADDR	Identifies the 24 most-significant address bits of the base of local access window $n$ . The specified base address should be aligned to the window size, as defined by LAWAR $n$ [SIZE].

## 2.2.3.5 Local Access Window $n$ Attributes Registers (LAWAR0–LAWAR9)

Figure 2-5 shows the bit fields of the LAWAR $n$  registers.



**Figure 2-5. Local Access Window  $n$  Attributes Registers (LAWAR0–LAWAR7)**

Table 2-8 describes LAWAR $n$  fields.

**Table 2-8. LAWAR $n$  Field Descriptions**

Bits	Name	Description
0	EN	0 The local access window $n$ (and all other LAWAR $n$ and LAWBAR $n$ fields) are disabled. 1 The local access window $n$ is enabled and other LAWAR $n$ and LAWBAR $n$ fields combine to identify an address range for this window.
1–7	—	Write reserved, read = 0

Table 2-8. LAWAR $n$  Field Descriptions (continued)

Bits	Name	Description
8–11	TRGT_ID	Identifies the target interface ID when a transaction hits in the address range defined by this window. Note that configuration registers and SRAM regions are mapped by the windows defined by CCSRBAR and L2SRBAR. These mappings supersede local access window mappings, so configuration registers and SRAM do not appear as a target for local access windows.  0000 PCI 0001 PCI Express 2 0010 PCI Express 1 0011 PCI Express 3 0100 Local bus memory controller 0101–1011Reserved 1100 Reserved 1101–1110Reserved 1111 DDR SDRAM
12–25	—	Write reserved, read = 0
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010Reserved 001011 4 Kbytes 001100 8 Kbytes 00110116 Kbytes ..... $2^{(SIZE+1)}$ bytes 100010 32 Gbytes 100011–111111Reserved

### 2.2.3.6 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence. For instance, if two windows are set up as shown in Table 2-9, local access window 1 governs the mapping of the 1-Mbyte region from 0x0\_7FF0\_0000 to 0x0\_7FFF\_FFF, even though the window described in local access window 2 also encompasses that memory region.

Table 2-9. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x0_7FF0_0000	1 Mbyte	0b0100 (Local bus controller —LBC)
2	0x0_0000_0000	2 Gbytes	0b1111 (DDR SDRAM)

### 2.2.3.7 Configuring Local Access Windows

Once a local access window is enabled, it should not be modified while any device in the system may be using the window. Neither should a new window be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last local access window configuration register before enabling any other devices to use the window. For instance, if local access windows 0–3 are being configured in order during the initialization process, the last write (to LAWAR3) should be followed by a read of LAWAR3 before any devices try to use any of these windows. If the configuration is being done by the local e500 processor, the read of LAWAR3 should be followed by an **isync** instruction.



### 2.2.3.8 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the local access windows and the additional mapping functions that happen at the target interface. The local access windows define how a transaction is routed through the MPC8533E internal interconnects from the transactions source to its target. After the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR SDRAM controller has chip select registers that map a memory request to a particular external device. Similarly, the local bus controller has base registers that perform a similar function. The PCI and PCI Express interfaces have outbound address translation and mapping units that map the local address into an external address space.

These other mapping functions are configured by programming the configuration, control, and status registers of the individual interfaces. Note that there is no need to have a one-to-one correspondence between local access windows and chip select regions or outbound ATMU windows. A single local access window can be further decoded to any number of chip selects or to any number or outbound ATMU windows at the target interface.

### 2.2.3.9 Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects

If a local access window maps an address to an interface other than the DDR SDRAM controller, then there should not be a valid chip select configured for the same address in the DDR SDRAM controller. Because DDR SDRAM chip selects boundaries are defined by a beginning and ending address, it is easy to define them so that they do not overlap with local access windows that map to other interfaces.

## 2.2.4 Outbound Address Translation and Mapping Windows

Outbound address translation and mapping refers to the translation of addresses from the local 36-bit address space to the external address space and attributes of a particular I/O interface. On the MPC8533E, the following blocks have outbound address translation and mapping units (ATMUs):

- PCI
- PCI Express

The PCI controller has four outbound ATMU windows plus a default window. The PCI outbound ATMU registers include extended translation address registers so that up to 64 bits of external address space can be supported. See [Section 17.3.1.2, “PCI ATMU Outbound Registers,”](#) for a detailed description of the PCI outbound ATMU windows.

The PCI Express interface has four outbound ATMU windows plus a default window. The PCI Express outbound ATMU registers include an extended translation address register so that up to 64 bits of external address space can be supported. See [Section 18.3.5.1, “PCI Express Outbound ATMU Registers”](#) for a detailed description of the PCI Express outbound ATMU windows.

## 2.2.5 Inbound Address Translation and Mapping Windows

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface (such as PCI address space) to the local 36-bit address space understood by the

internal interfaces of the MPC8533E. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. The PCI and the PCI Express controllers have inbound address translation and mapping units (ATMUs).

### 2.2.5.1 PCI Inbound ATMU

The PCI controller has three general inbound ATMU windows plus a dedicated window for memory mapped configuration accesses (PCSRBAR). These windows have a one-to-one correspondence with the base address registers in the PCI programming model. Updating one automatically updates the other. There is no default inbound window; if a PCI address does not match one of the inbound ATMU windows, the MPC8533E does not respond with an assertion of `PCI_DEVSEL`. See [Section 17.3.1.3, “PCI ATMU Inbound Registers,”](#) for a detailed description of the PCI inbound ATMU windows.

### 2.2.5.2 PCI Express Inbound ATMU

The PCI Express controller has three inbound ATMU windows plus a default. See [Section 18.3.5.1, “PCI Express Outbound ATMU Registers,”](#) for a detailed description of the PCI Express inbound ATMU windows.

### 2.2.5.3 Illegal Interaction Between Inbound ATMUs and Local Access Windows

Since both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another. For instance, it is a programming error to have an inbound ATMU map a transaction to the DDR SDRAM memory controller (target interface 0b1111) if the resulting translated local address is mapped to PCI (target interface 0b0000) by a local access window. Such a programming error may result in unpredictable system deadlocks.

## 2.3 Configuration, Control, and Status Register Map

All of the memory mapped configuration, control, and status registers in the MPC8533E are contained within a 1-Mbyte address region. To allow for flexibility, the configuration, control, and status block is relocatable in the local address space. The local address map location of this register block is controlled by the configuration, control, and status registers base address register (CCSRBAR), see [Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register \(CCSRBAR\).”](#) The default value for CCSRBAR is 4 Gbytes–9 Mbytes, or 0x0\_FF70\_0000.

#### NOTE

The configuration, control, and status window must not overlap a local access window that maps to the DDR controller. Otherwise, undefined behavior occurs.

### 2.3.1 Accessing CCSR Memory from the Local Processor

When the local e500 processor is used to configure CCSR space, the CCSR memory space should typically be marked as cache-inhibited and guarded.

In addition, many configuration registers affect accesses to other memory regions; therefore writes to these registers must be guaranteed to have taken effect before accesses are made to the associated memory regions.

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be chased by a read of the same register, and that should be followed by a SYNC instruction. Then accesses can safely be made to memory regions affected by the configuration register write.

### 2.3.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the e500 processor, the configuration, control, and status registers are accessible from external interfaces. This allows external masters on the I/O ports to configure the MPC8533E.

External masters do not need to know the location of the CCSR memory in the local address map. Rather, they access this region of the local memory map through a window defined by a register in the interface programming model that is accessible to the external master from its external memory map.

The PCI base address for accessing the local CCSR memory is selectable through the PCI configuration and status register base address register (PCSRBAR), at offset 0x10, described in [Section 17.3.2.11, “PCI Base Address Registers.”](#) An external PCI master sets this register by running a PCI configuration cycle to the MPC8533E. Subsequent memory accesses by a PCI master to the PCI address range indicated by PCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

### 2.3.3 Organization of CCSR Memory

The configuration, control, and status registers of the MPC8533E are grouped according to functional units. Most functional blocks are allocated a 4-Kbyte address space for registers. Registers that fall into this category are referred to as general utilities registers. These registers occupy the first 256 Kbytes of CCSR memory.

Registers controlling functions that are not particular to a functional unit but to the device as a whole occupy the highest 256 Kbytes of CCSR memory and are referred to as device-specific registers.

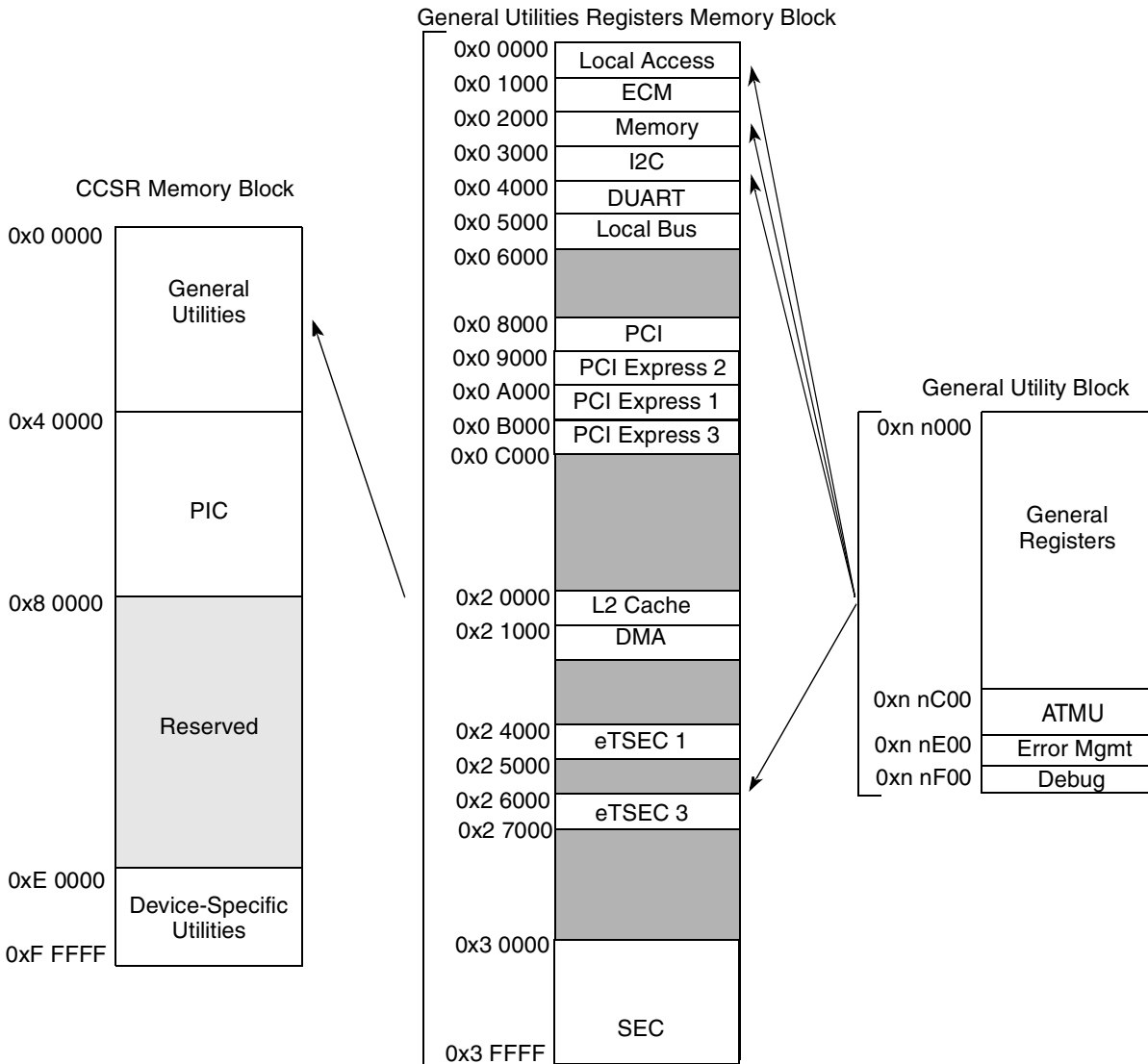
Some functional units, such as the OpenPIC-based interrupt controller have larger address spaces as defined by their programming models. The registers for these blocks are given their own large regions of CCSR memory.

**Table 2-10. Local Memory Configuration, Control, and Status Register Summary**

Offset from CCSRBAR	Register Grouping
0x0_0000–0x3_FFFF	General utilities
0x4_0000–0x7_FFFF	Programmable interrupt controller (PIC)
0x8_0000–0xB_FFFF	Reserved
0xC_0000–0xD_FFFF	Reserved
0xE_0000–0xF_FFFF	Device-specific utilities

### 2.3.4 General Utilities Registers

Figure 2-6 provides an overview of the general utilities registers.



**Figure 2-6. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block**

Figure 2-6 also shows the organization of registers inside the 4-Kbyte register space allocated to an individual functional block. The first 3 Kbytes are available for general registers. The next 512 bytes are dedicated to address translation and mapping registers, if applicable to that particular functional unit (for example, PCI). If a unit has error management registers, they are typically placed starting at offset 0xE00 from the beginning of the block’s 4-Kbyte space, and any debug registers are typically placed in the final 256 bytes of the unit’s register space starting at offset 0xF00.

General utilities registers are accessed as 32-bit quantities except for the DUART and I<sup>2</sup>C registers, which are accessed as bytes.

### NOTE

Refer to detailed register descriptions for each functional unit for exact locations, sizes, and access requirements. Some blocks may have exceptions to the above guidelines.

## 2.3.5 Interrupt Controller and CCSR

The programmable interrupt controller (PIC) registers are at offset 0x4\_0000 from CCSRBAR, see [Figure 2-7](#). Its programming model follows the OpenPIC architecture. The interrupt controller registers should only be accessed with 32-bit accesses.

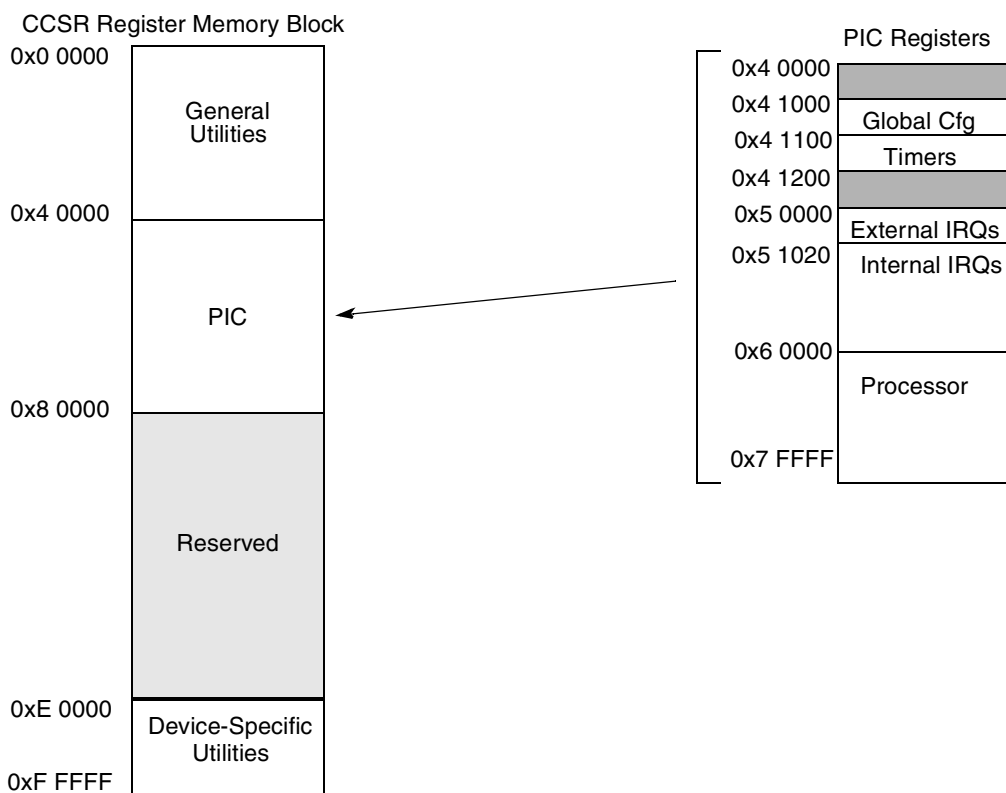
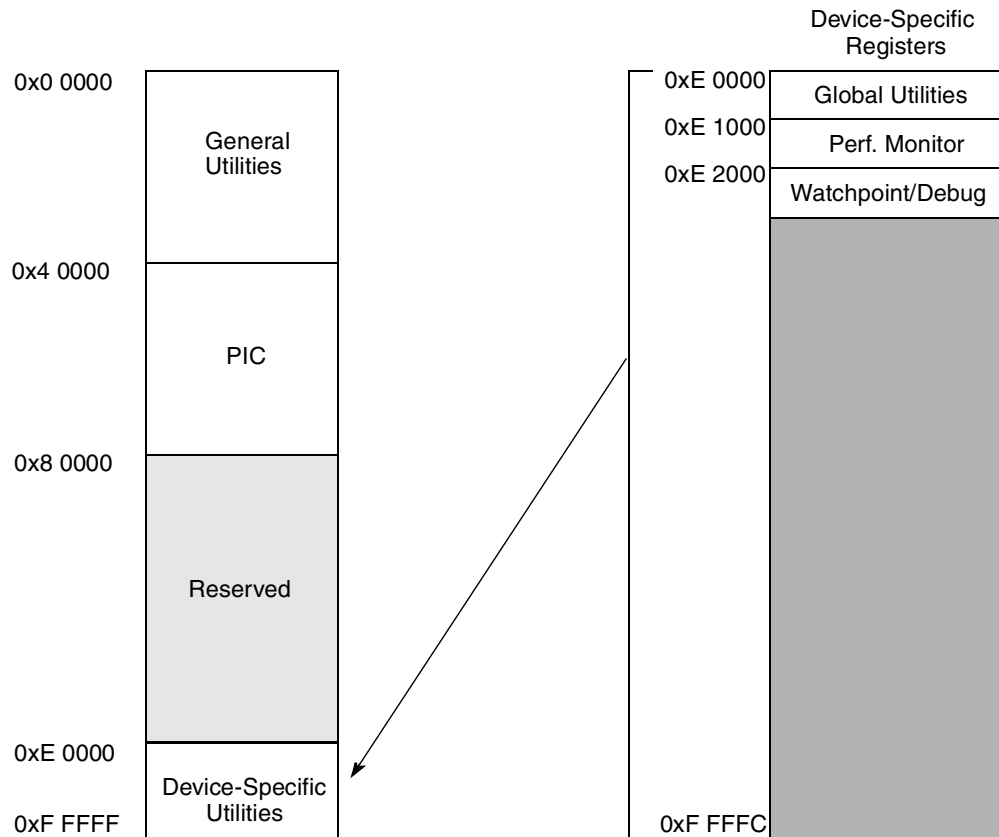


Figure 2-7. PIC Mapping to Configuration, Control, and Status Memory Block

## 2.3.6 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities (refer to [Figure 2-8](#)). These registers are accessible with 32-bit accesses only. Transactions of other than 32-bit are considered a programming error and operation is undefined.

Reserved bits in the following register descriptions are not guaranteed to have predictable values. Software must preserve the values of reserved bits when writing to a register. Also, when reading from a register, software should not rely on the value of any reserved bit remaining consistent.



**Figure 2-8. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block**

## 2.4 Complete CCSR Map

Table 2-11 lists the MPC8533E memory-mapped registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

### NOTE

In the eTSEC (three-speed Ethernet controller) section of the following table, registers denoted with an asterisk (\*) are new to the enhanced TSEC (eTSEC) and are not supported by PowerQUICC III TSECs.

Table 2-11. CCSR Block Base Address Map

Block Base Address (Hex)	Block	Section/Page	Comments
<b>General Utilities (0x0_0000–0x3_FFFF)</b>			
0x0_0000	Local access registers	4.3.1/4-4	0x0_0000: Configuration, control, and status registers 0x0_0BF8: Local access window base and size registers
0x0_1000	e500 coherency module (ECM)	8.2/8-3	
0x0_2000	DDR memory controller	9.4/9-9	
0x0_3000	I <sup>2</sup> C controllers (2)	10.3/10-9	I <sup>2</sup> C controller 1: 0x0_3000 I <sup>2</sup> C controller 2: 0x0_3100
0x0_4000	DUART	13.3/13-4	UART0: 0x0_4500 UART1: 0x0_4600
0x0_5000	Local bus controller (LBC)	14.3/14-8	
0x0_6000– 0x0_7FFF	Reserved		
0x0_8000	PCI controllers	17.3/17-11	
0x0_9000	PCI Express controller 2	18.3/18-5	PCI Express controller 1 is located at 0x0_A000. PCI Express controller 3 is located at 0x0_B000.
0x0_A000	PCI Express controller 1	18.3/18-5	PCI Express controller 2 is located at 0x0_9000. PCI Express controller 3 is located at 0x0_B000.
0x0_B000	PCI Express controller 3	18.3/18-5	PCI Express controller 1 is located at 0x0_A000. PCI Express Controller 2 is located at 0x0_9000.
0x0_C000– 0x1_FFFF	Reserved		
0x2_0000	L2/SRAM memory-mapped configuration	7.3/7-8	
0x2_1000	DMA controller	16.3/16-6	DMA0: 0x2_1100 DMA1: 0x2_1180 DMA2: 0x2_1200 DMA3: 0x2_1280 General Status: 0x2_1300
0x2_2000– 0x2_3FFF	Reserved		
0x2_4000	eTSEC1	15.5/15-12	
0x2_5000	Reserved		
0x2_6000	eTSEC3	15.5/15-12	
0x2_7000– 0x3_0FFF	Reserved		

Table 2-11. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section/Page	Comments
0x3_1000– 0x3_1004	Reserved		
0x3_1008	Integrated security engine	<a href="#">12.2/12-10</a>	
0x3_100C– 0x3_FFFF	Reserved		
<b>Programmable Interrupt Controller (PIC) (0x4_0000–0x7_FFFF)</b>			
0x4_0000	PIC—Global registers	<a href="#">10.3.1/10-18</a>	
0x5_0000	PIC—Interrupt source configuration registers	<a href="#">10.3.7/10-39</a>	
0x6_0000	PIC— Per-CPU registers	<a href="#">10.3.8/10-44</a>	
0x7_0000– 0x7_FFFF	Reserved		
<b>Reserved (0x8_0000–0xD_FFFF)</b>			
<b>Device Specific Utilities (0xE_0000–0xF_FFFF)</b>			
0xE_0000	Global Utilities	<a href="#">19.4/19-3</a>	
0xE_1000	Performance Monitor	<a href="#">20.3/20-3</a>	
0xE_2000	Watchpoint Monitor and Trace Buffer	<a href="#">21.3/21-9</a>	
0xE_3000– 0xF_FFFF	Reserved		



## Chapter 3

# Signal Descriptions

This chapter describes the MPC8533E external signals. It is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions, including two lists: one by functional block and one alphabetical
- List of reset configuration signals
- List of output signal states at reset

### NOTE

A bar over a signal name indicates that the signal is active low, such as  $\overline{\text{IRQ\_OUT}}$  (interrupt out). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as IRQ (interrupt input), are referred to as asserted when they are high and negated when they are low.

Internal signals are shown throughout this document as lower case and in italics. For example, *sys\_logic\_clk* is an internal signal. These are discussed only as necessary for understanding the external functionality of the device.

## 3.1 Signals Overview

The MPC8533E signals are grouped as follows:

- PCI interface signals
- I<sup>2</sup>C interface signals
- System control, general-purpose input/output, power management, and debug signals
- Test, JTAG, configuration, and clock signals
- PCI Express 1, 2, and 3 interface signals
- eTSEC 1 and 3 interface signals
- DDR memory interface signals
- Local bus interface signals
- DMA interface signals
- PIC interface signals
- DUART interface signals
- Configuration signals

Figure 3-1, Figure 3-2, and Figure 3-3 illustrate the external signals of the MPC8533E, showing how the signals are grouped. Refer to the *MPC8533E Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications. Note that these figures show multiplexed signals multiple times.

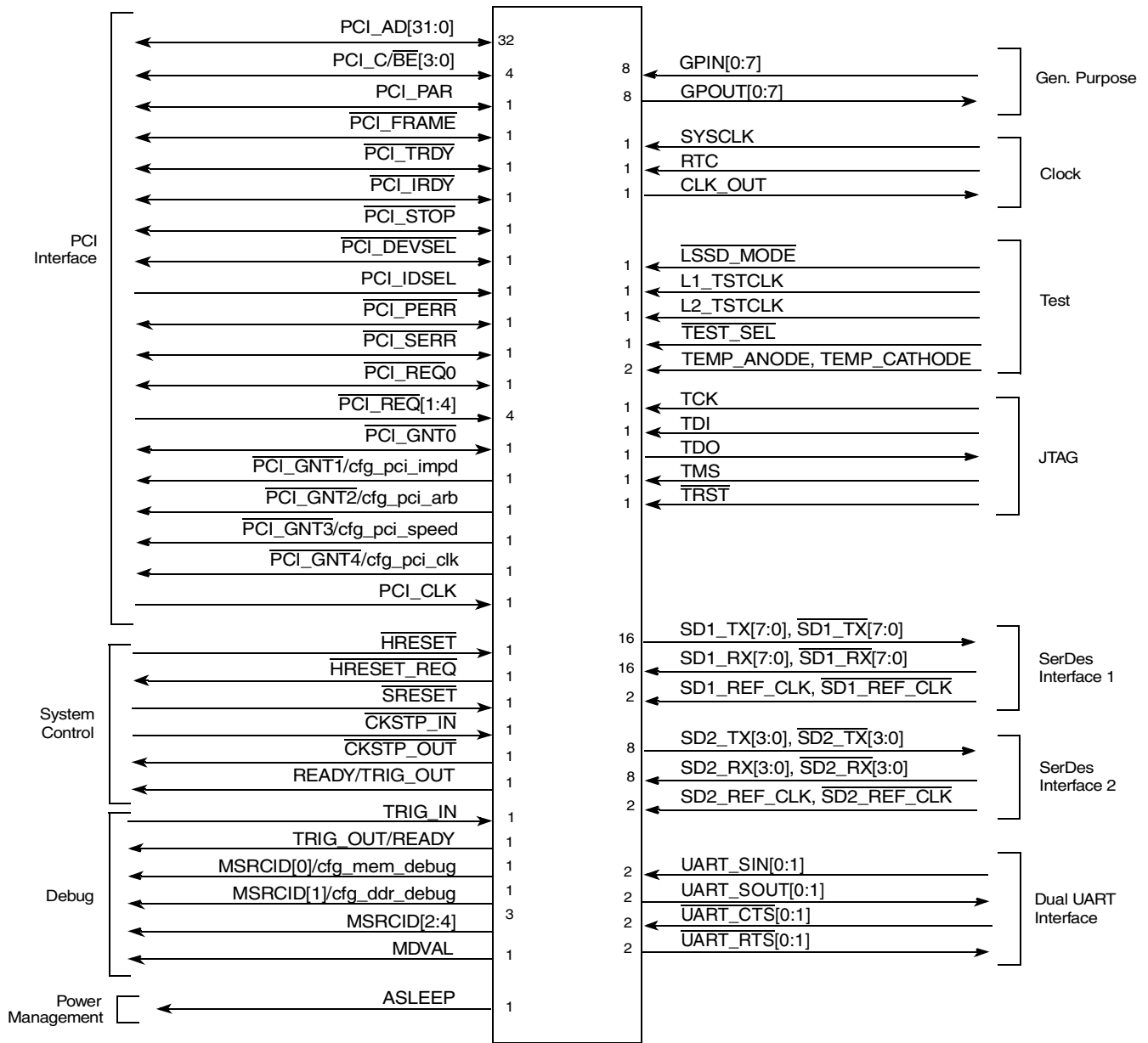


Figure 3-1. MPC8533E Signal Groupings (1/3)

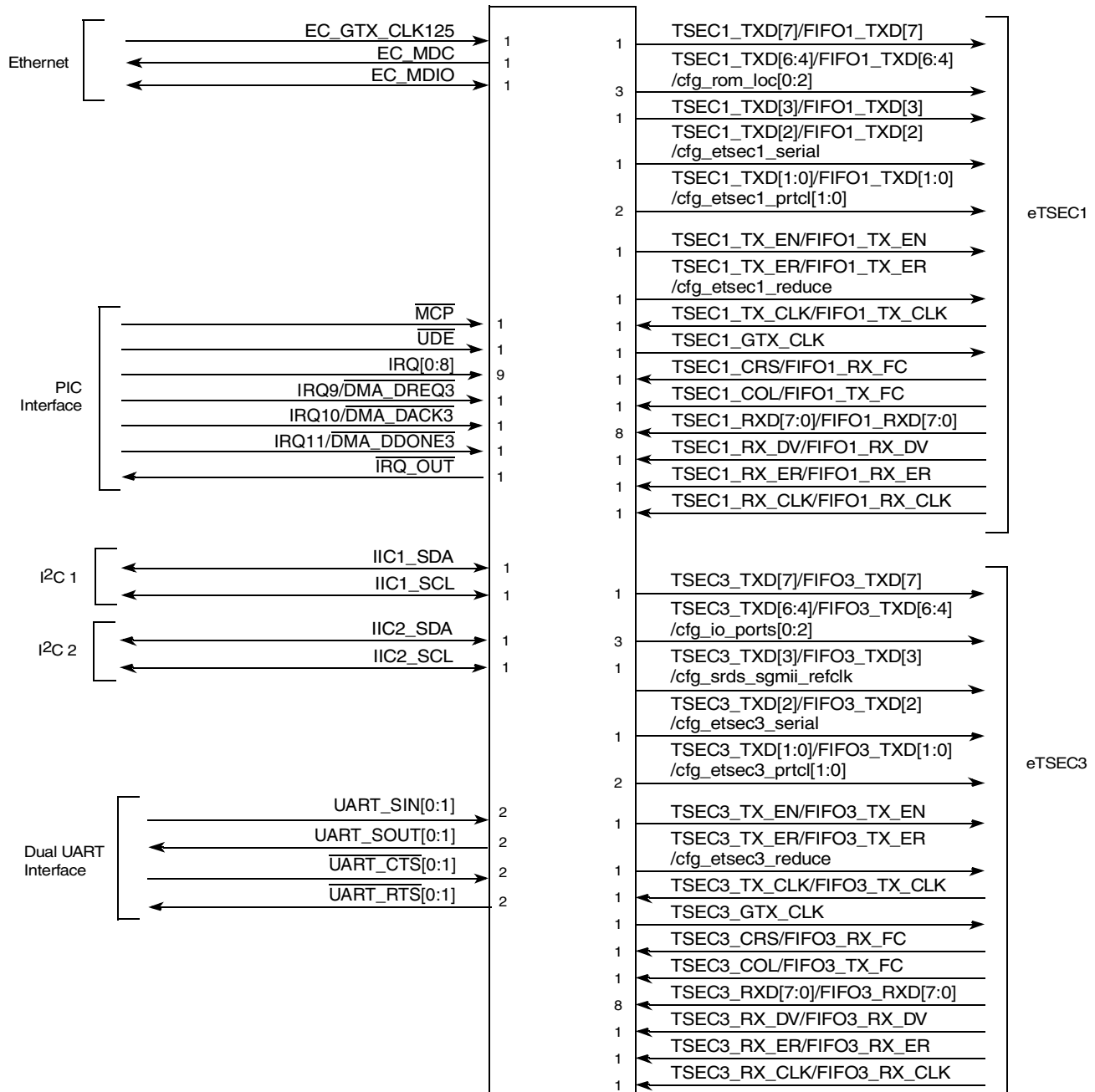


Figure 3-2. MPC8533E Signal Groupings (2/3) (Continued)

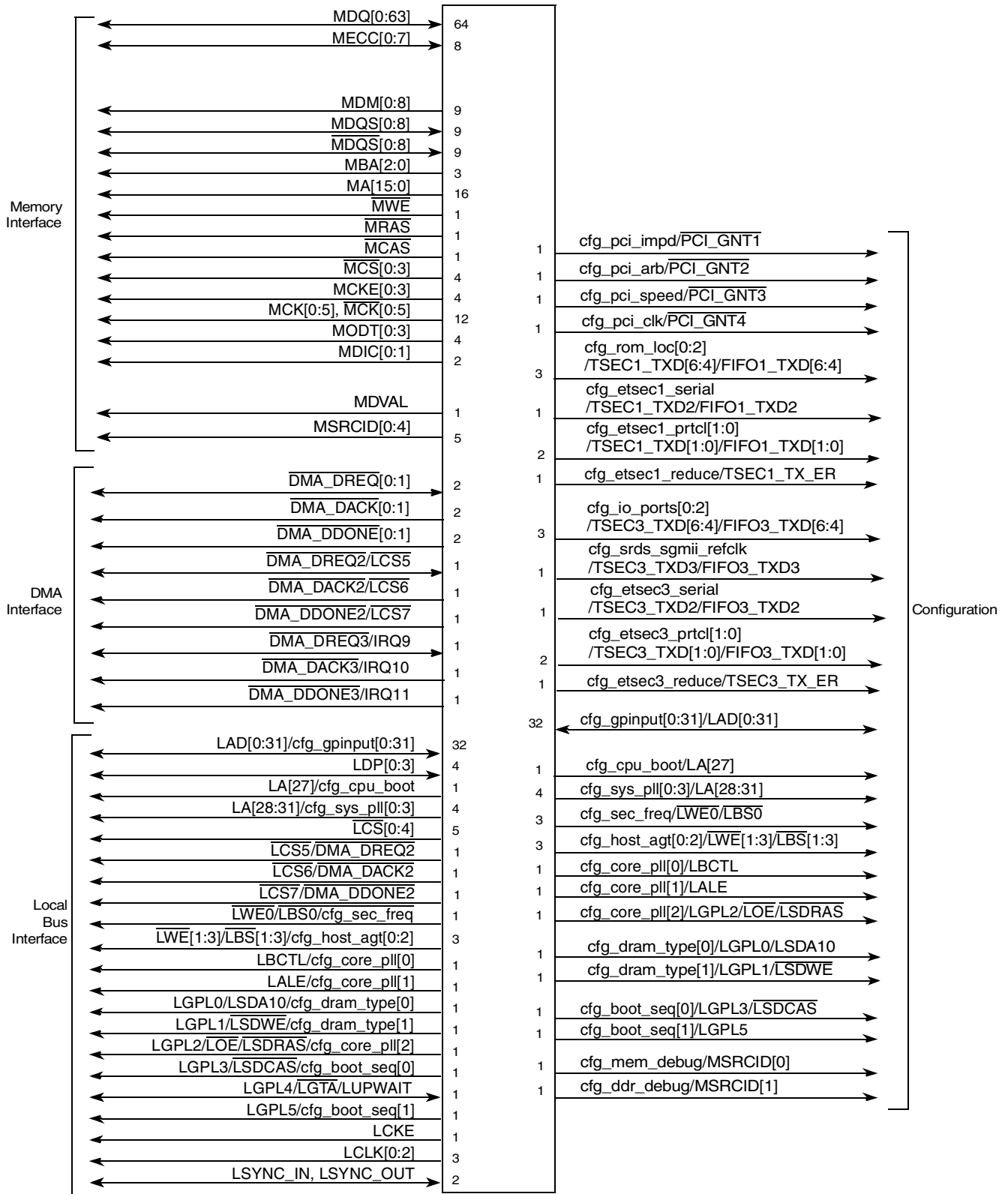


Figure 3-3. MPC8533E Signal Groupings (3/3) (Continued)

Note that individual chapters of this document provide details for each signal, describing each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

The following tables provide summaries of signal functions. [Table 3-1](#) lists signals grouped by function, and [Table 3-2](#) lists signals alphabetically. These tables detail the signal name, interface, alternate functions, number of signals, and whether the signal is an input, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the tables provide a pointer to the table where the signal function is described.

**Table 3-1. MPC8533E Signal Reference by Functional Block**

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	<a href="#">9-3/9-5</a>
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	<a href="#">9-3/9-5</a>
MDM[0:7]	DDR data mask	DDR memory	—	8	O	<a href="#">9-3/9-5</a>
MDM8	DDR ECC data mask	DDR memory	—	1	O	<a href="#">9-3/9-5</a>
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	<a href="#">9-3/9-5</a>
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	<a href="#">9-3/9-5</a>
$\overline{\text{MDQS}}$ [0:8]	DDR ECC data strobe (complement)	DDR memory	—	9	I/O	<a href="#">9-3/9-5</a>
MBA[2:0]	DDR bank select	DDR memory	—	3	O	<a href="#">9-3/9-5</a>
MA[15:0]	DDR address	DDR memory	—	16	O	<a href="#">9-3/9-5</a>
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	<a href="#">9-3/9-5</a>
$\overline{\text{MRAS}}$	DDR row address strobe	DDR memory	—	1	O	<a href="#">9-3/9-5</a>
$\overline{\text{MCAS}}$	DDR column address strobe	DDR memory	—	1	O	<a href="#">9-3/9-5</a>
$\overline{\text{MCS}}$ [0:3]	DDR chip select (2/DIMM)	DDR memory	—	4	O	<a href="#">9-3/9-5</a>
MCKE[0:3]	DDR clock enable	DDR memory	—	4	O	<a href="#">9-3/9-5</a>
MCK[0:5], $\overline{\text{MCK}}$ [0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	<a href="#">9-3/9-5</a>
MODT[0:3]	DRAM On-Die Termination	DDR memory	—	4	O	<a href="#">9-3/9-5</a>
MDIC[0:1]	Driver impedance calibration	Debug/ DDR memory	—	2	I/O	<a href="#">9-3/9-5</a>
PCI_AD[31:0]	PCI address/data	PCI	—	32	I/O	<a href="#">17-2/17-6</a>
PCI_C/ $\overline{\text{BE}}$ [3:0]	PCI command/byte enable	PCI	—	4	I/O	<a href="#">17-2/17-6</a>
PCI_PAR	PCI parity	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
$\overline{\text{PCI\_FRAME}}$	PCI frame	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
$\overline{\text{PCI\_TRDY}}$	PCI target ready	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
$\overline{\text{PCI\_IRDY}}$	PCI initiator ready	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
$\overline{\text{PCI\_STOP}}$	PCI stop	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
$\overline{\text{PCI\_DEVSEL}}$	PCI device select	PCI	—	1	I/O	<a href="#">17-2/17-6</a>
PCI_IDSEL	PCI initial device select	PCI	—	1	I	<a href="#">17-2/17-6</a>

Table 3-1. MPC8533E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{PCI\_PERR}}$	PCI parity error	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_SERR}}$	PCI system error	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_REQ0}}$	PCI request 0	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_REQ}}[4:1]$	PCI request 4–1	PCI	—	4	I	17-2/17-6
$\overline{\text{PCI\_GNT0}}$	PCI grant 0	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_GNT1}}$	PCI grant 1	PCI	cfg_pci_impd	1	O	17-2/17-6
$\overline{\text{PCI\_GNT2}}$	PCI grant 2	PCI	cfg_pci_arb	1	O	17-2/17-6
$\overline{\text{PCI\_GNT3}}$	PCI grant 3	PCI	cfg_pci_speed	1	O	17-2/17-6
$\overline{\text{PCI\_GNT4}}$	PCI grant 4	PCI	cfg_pci_clk	1	O	17-2/17-6
PCI_CLK	PCI clock	PCI	—	1	I	17-2/17-6
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	17-2/17-6
EC_MDC	Ethernet management data clock	Ethernet management	—	1	O	17-2/17-6
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	17-2/17-6
TSEC1_TXD7	TSEC1 transmit data 7	eTSEC1	FIFO1_TXD7	1	O	15-2/15-9
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	eTSEC1	FIFO1_TXD[6:4] /cfg_rom_loc[0:2]	3	O	15-2/15-9
TSEC1_TXD3	TSEC1 transmit data 3	eTSEC1	FIFO1_TXD3	1	O	15-2/15-9
TSEC1_TXD2	TSEC1 transmit data 2	eTSEC1	FIFO1_TXD2 /cfg_etsec1_serial	1	O	15-2/15-9
TSEC1_TXD1	TSEC1 transmit data 1	eTSEC1	FIFO1_TXD1 /cfg_etsec1_prctl[1]	1	O	15-2/15-9
TSEC1_TXD0	TSEC1 transmit data 0	eTSEC1	FIFO1_TXD0 /cfg_etsec1_prctl[0]	1	O	15-2/15-9
TSEC1_TX_EN	TSEC1 transmit enable	eTSEC1	FIFO1_TX_EN	1	O	15-2/15-9
TSEC1_TX_ER	TSEC1 transmit error	eTSEC1	FIFO1_TX_ER /cfg_etsec1_reduce	1	O	15-2/15-9
TSEC1_TX_CLK	TSEC1 transmit clock in	eTSEC1	FIFO1_TX_CLK	1	I	15-2/15-9
TSEC1_GTX_CLK	TSEC1 transmit clock out	eTSEC1	—	1	O	15-2/15-9
TSEC1_CRS	TSEC1 carrier sense	eTSEC1	FIFO1_RX_FC	1	I/O	15-2/15-9
TSEC1_COL	TSEC1 collision detect	eTSEC1	FIFO1_TX_FC	1	I	15-2/15-9
TSEC1_RXD[7:0]	TSEC1 receive data	eTSEC1	FIFO1_RXD[7:0]	8	I	15-2/15-9
TSEC1_RX_DV	TSEC1 receive data valid	eTSEC1	FIFO1_RX_DV	1	I	15-2/15-9
TSEC1_RX_ER	TSEC1 receiver error	eTSEC1	FIFO1_RX_ER	1	I	15-2/15-9
TSEC1_RX_CLK	TSEC1 receive clock	eTSEC1	FIFO1_RX_CLK	1	I	15-2/15-9
TSEC3_TXD7	TSEC3 transmit data 7	eTSEC3	FIFO3_TXD7	1	O	15-2/15-9

Table 3-1. MPC8533E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC3_TXD[6:4]	TSEC3 transmit data [6:4]	eTSEC3	FIFO3_TXD[6:4] /cfg_io_ports[0:2]	3	O	15-2/15-9
TSEC3_TXD3	TSEC3 transmit data 3	eTSEC3	FIFO3_TXD3	1	O	15-2/15-9
TSEC3_TXD2	TSEC3 transmit data 2	eTSEC3	FIFO3_TXD2 cfg_etsec3_serial	1	O	15-2/15-9
TSEC3_TXD[1:0]	TSEC3 transmit data 1–0	eTSEC3	FIFO3_TXD[1:0] cfg_etsec3_prctl[1:0]	2	O	15-2/15-9
TSEC3_TX_EN	TSEC3 transmit enable	eTSEC3	FIFO3_TX_EN	1	O	15-2/15-9
TSEC3_TX_ER	TSEC3 transmit error	eTSEC3	FIFO3_TX_ER cfg_etsec3_reduce	1	O	15-2/15-9
TSEC3_TX_CLK	TSEC3 transmit clock in	eTSEC3	FIFO3_TX_CLK	1	I	15-2/15-9
TSEC3_GTX_CLK	TSEC3 transmit clock out	eTSEC3	—	1	O	15-2/15-9
TSEC3_CRS	TSEC3 carrier sense	eTSEC3	FIFO3_RX_FC	1	I/O	15-2/15-9
TSEC3_COL	TSEC3 collision detect	eTSEC3	FIFO3_TX_FC	1	I	15-2/15-9
TSEC3_RXD[7:0]	TSEC3 receive data 7–0	eTSEC3	FIFO3_RXD[7:0]	8	I	15-2/15-9
TSEC3_RX_DV	TSEC3 receive data valid	eTSEC3	FIFO3_RX_DV	1	I	15-2/15-9
TSEC3_RX_ER	TSEC3 receive error	eTSEC3	FIFO3_RX_ER	1	I	15-2/15-9
TSEC3_RX_CLK	TSEC3 receive clock	eTSEC3	FIFO3_RX_CLK	1	I	15-2/15-9
LAD[0:31]	Local bus address/data	LBC	cfg_gpinput[0:31]	32	I/O	14-2/14-5
LDP[0:3]	Local bus data parity	LBC	—	4	I/O	14-2/14-5
LA27	Local bus burst address	LBC	cfg_cpu_boot	1	O	14-2/14-5
LA[28:31]	Local bus port address	LBC	cfg_sys_pll[0:3]	4	O	14-2/14-5
$\overline{\text{LCS}}[0:4]$	Local bus chip select 0–4	LBC	—	5	O	14-2/14-5
$\overline{\text{LCS}}5$	Local bus chip select 5	LBC	$\overline{\text{DMA\_DREQ}}2$	1	I/O	14-2/14-5
$\overline{\text{LCS}}6$	Local bus chip select 6	LBC	$\overline{\text{DMA\_DACK}}2$	1	O	14-2/14-5
$\overline{\text{LCS}}7$	Local bus chip select 7	LBC	$\overline{\text{DMA\_DDONE}}2$	1	O	14-2/14-5
$\overline{\text{LWE}}0$ /LBS0	Local bus write enable /byte select 0	LBC	—	1	O	14-2/14-5
$\overline{\text{LWE}}[1:3]$ / $\overline{\text{LBS}}[1:3]$	Local bus write enable /byte select 1–3	LBC	cfg_host_agt[0:2]	3	O	14-2/14-5
LBCTL	Local bus data buffer control	LBC	cfg_core_pll[0]	1	O	14-2/14-5
LALE	Local bus address latch enable	LBC	cfg_core_pll[1]	1	O	14-2/14-5
LGPL0	Local bus UPM general purpose line 0	LBC	cfg_dram_type[0]	1	O	14-2/14-5
LGPL1	Local bus GP line 1	LBC	cfg_dram_type[1]	1	O	14-2/14-5
LGPL2 /LOE	Local bus GP line 2 /output enable	LBC	cfg_core_pll[2]	1	O	14-2/14-5

Table 3-1. MPC8533E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
LGPL3	Local bus GP line 3	LBC	cfg_boot_seq[0]	1	O	14-2/14-5
LGPL4 /LGTA/ LUPWAIT	Local bus GP line 4 /GPCM terminate access /UPM wait	LBC	—	1	I/O	14-2/14-5
LGPL5	Local bus GP line 5 address	LBC	cfg_boot_seq[1]	1	O	14-2/14-5
LCKE	Local bus clock enable	LBC	—	1	O	14-2/14-5
LCLK[0:2]	Local bus clock	LBC	—	3	O	14-2/14-5
LSYNC_IN	Local bus DLL synchronization	LBC	—	1	I	14-2/14-5
LSYNC_OUT	Local bus DLL synchronization	LBC	—	1	O	14-2/14-5
$\overline{\text{DMA\_DREQ}}[0:1]$	DMA request 0–1	DMA	—	2	I	16-3/16-6
$\overline{\text{DMA\_DREQ}}2$	DMA request 2	DMA	$\overline{\text{LCS}}5$	1	I	16-3/16-6
$\overline{\text{DMA\_DREQ}}3$	DMA request 3	DMA	IRQ9	1	I	16-3/16-6
$\overline{\text{DMA\_DACK}}[0:1]$	DMA acknowledge 0–1	DMA	—	2	O	16-3/16-6
$\overline{\text{DMA\_DACK}}2$	DMA acknowledge 2	DMA	$\overline{\text{LCS}}6$	1	O	16-3/16-6
$\overline{\text{DMA\_DACK}}3$	DMA acknowledge 3	DMA	IRQ10	1	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}[0:1]$	DMA done 0–1	DMA	—	2	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}2$	DMA done 2	DMA	$\overline{\text{LCS}}7$	1	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}3$	DMA done 3	DMA	IRQ11	1	O	16-3/16-6
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-5/10-8
$\overline{\text{UDE}}$	Unconditional debug event	PIC	—	1	I	10-5/10-8
IRQ[0:8]	External interrupt 0–8	PIC	—	9	I	10-5/10-8
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ}}3$	1	I	10-5/10-8
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK}}3$	1	I	10-5/10-8
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE}}3$	1	I	10-5/10-8
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-5/10-8
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	13-2/13-3
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	13-2/13-3
$\overline{\text{UART\_CTS}}[0:1]$	DUART clear to send	Dual UART	—	2	I	13-2/13-3
$\overline{\text{UART\_RTS}}[0:1]$	DUART ready to send	Dual UART	—	2	O	13-2/13-3
IIC1_SDA	I <sup>2</sup> C serial data	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC1_SCL	I <sup>2</sup> C serial clock	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC2_SDA	I <sup>2</sup> C serial data	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC2_SCL	I <sup>2</sup> C serial clock	I <sup>2</sup> C	—	1	I/O	11-2/11-4
SD1_RX[3:0], SD1_RX[3:0]	Receive data, receive data complement	PCI Express 1	—	8	I	18-2/18-5



Table 3-1. MPC8533E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
SD1_RX[7:4], SD1_RX[7:4]	Receive data, receive data complement	PCI Express 2	—	8	I	18-2/18-5
SD1_TX[3:0], SD1_TX[3:0]	Transmit data, transmit data complement	PCI Express 1	—	8	O	18-2/18-5
SD1_TX[7:4], SD1_TX[7:4]	Transmit data, transmit data complement	PCI Express 2	—	8	O	18-2/18-5
SD1_REF_CLK, SD1_REF_CLK	SerDes1 reference clock, SerDes1 reference clock complement	PCI Express 1, PCI Express 2	—	2	I	18-2/18-5
SD2_RX[0], SD2_RX[0]	Receive data, receive data complement	PCI Express 3	—	4	I	18-2/18-5
SD2_TX[0], SD2_TX[0]	Transmit data, transmit data complement	PCI Express 3	—	4	O	18-2/18-5
SD2_REF_CLK, SD2_REF_CLK	SerDes2 reference clock, SerDes2 reference clock complement	PCI Express 3	—	2	I	18-2/18-5
HRESET	Hard reset	System control	—	1	I	4-2/4-2
HRESET_REQ	Hard reset request	System control	—	1	O	4-2/4-2
SRESET	Soft reset	System control	—	1	I	4-2/4-2
CKSTP_IN	Checkstop in	System control	—	1	I	19-2/19-2
CKSTP_OUT	Checkstop out	System control	—	1	O	19-2/19-2
GPOUT[0:7]	General-purpose output	General-purpose outputs	—	8	O	19-2/19-2 19.4.1.9/19-12
GPIN[0:7]	General-purpose input	General-purpose inputs	—	8	I	19-2/19-2 19.4.1.9/19-12
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
ASLEEP	Asleep	Power mgmt	—	1	O	19-2/19-2
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	21-4/21-7
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	21-4/21-7
MSRCID0	Memory debug source port ID 0	Debug	cfg_mem_debug	1	O	21-3/21-6
MSRCID1	Memory debug source port ID 1	Debug	cfg_ddr_debug	1	O	21-3/21-6
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	21-3/21-6
MDVAL	Memory debug data valid	Debug	—	1	O	21-3/21-6
LSSD_MODE	LSSD mode	Test	—	1	I	21-5/21-8
L1_TSTCLK	L1 test clock	Test	—	1	I	21-5/21-8
L2_TSTCLK	L2 test clock	Test	—	1	I	21-5/21-8
TEST_SEL	Test select	Test	—	1	I	21-5/21-8

Table 3-1. MPC8533E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TEMP_ANODE	Thermal diode access	Test	—	2	I	21-5/21-8
TEMP_CATHODE	Thermal diode access	Test	—	2	I	21-5/21-8
TCK	Test clock	JTAG	—	1	I	21-5/21-8
TDI	Test data in	JTAG	—	1	I	21-5/21-8
TDO	Test data out	JTAG	—	1	O	21-5/21-8
TMS	Test mode select	JTAG	—	1	I	21-5/21-8
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	21-5/21-8
SYCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
RTC	Real time clock	Clock	—	1	I	4-3/4-3
CLK_OUT	Clock out	Clock	—	1	O	19-2/19-2

Table 3-2. Alphabetical Signal Reference

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
ASLEEP	Asleep	Power mgmt	—	1	O	19-2/19-2
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	19-2/19-2
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	19-2/19-2
CLK_OUT	Clock out	Clock	—	1	O	19-2/19-2
$\overline{\text{DMA\_DACK}}[0:1]$	DMA acknowledge 0–1	DMA	—	2	O	16-3/16-6
$\overline{\text{DMA\_DACK}}2$	DMA acknowledge 2	DMA	$\overline{\text{LCS}}6$	1	O	16-3/16-6
$\overline{\text{DMA\_DACK}}3$	DMA acknowledge 3	DMA	IRQ10	1	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}[0:1]$	DMA done 0–1	DMA	—	2	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}2$	DMA done 2	DMA	$\overline{\text{LCS}}7$	1	O	16-3/16-6
$\overline{\text{DMA\_DDONE}}3$	DMA done 3	DMA	IRQ11	1	O	16-3/16-6
$\overline{\text{DMA\_DREQ}}[0:1]$	DMA request 0–1	DMA	—	2	I	16-3/16-6
$\overline{\text{DMA\_DREQ}}2$	DMA request 2	DMA	$\overline{\text{LCS}}5$	1	I	16-3/16-6
$\overline{\text{DMA\_DREQ}}3$	DMA request 3	DMA	IRQ9	1	I	16-3/16-6
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	17-2/17-6
EC_MDC	Ethernet management data clock	Ethernet management	—	1	O	17-2/17-6
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	17-2/17-6
GPIN[0:7]	General-purpose input	General-purpose inputs	—	8	I	19-2/19-2 19.4.1.9/19-12
GPOUT[0:7]	General-purpose output	General-purpose outputs	—	8	O	19-2/19-2 19.4.1.9/19-12

Table 3-2. Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-2/4-2
IIC1_SCL	I <sup>2</sup> C serial clock	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC1_SDA	I <sup>2</sup> C serial data	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC2_SCL	I <sup>2</sup> C serial clock	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IIC2_SDA	I <sup>2</sup> C serial data	I <sup>2</sup> C	—	1	I/O	11-2/11-4
IRQ[0:8]	External interrupt 0–8	PIC	—	9	I	10-5/10-8
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-5/10-8
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-5/10-8
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-5/10-8
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-5/10-8
L1_TSTCLK	L1 test clock	Test	—	1	I	21-5/21-8
L2_TSTCLK	L2 test clock	Test	—	1	I	21-5/21-8
LA[28:31]	Local bus port address	LBC	cfg_sys_pll[0:3]	4	O	14-2/14-5
LA27	Local bus burst address	LBC	cfg_cpu_boot	1	O	14-2/14-5
LAD[0:31]	Local bus address/data	LBC	cfg_gpinput[0:31]	32	I/O	14-2/14-5
LALE	Local bus address latch enable	LBC	cfg_core_pll[1]	1	O	14-2/14-5
LBCTL	Local bus data buffer control	LBC	cfg_core_pll[0]	1	O	14-2/14-5
LCKE	Local bus clock enable	LBC	—	1	O	14-2/14-5
LCLK[0:2]	Local bus clock	LBC	—	3	O	14-2/14-5
$\overline{\text{LCS}}[0:4]$	Local bus chip select 0–4	LBC	—	5	O	14-2/14-5
$\overline{\text{LCS5}}$	Local bus chip select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	I/O	14-2/14-5
$\overline{\text{LCS6}}$	Local bus chip select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	14-2/14-5
$\overline{\text{LCS7}}$	Local bus chip select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	14-2/14-5
LDP[0:3]	Local bus data parity	LBC	—	4	I/O	14-2/14-5
LGPL0	Local bus UPM general purpose line 0	LBC	cfg_dram_type[0]	1	O	14-2/14-5
LGPL1	Local bus GP line 1	LBC	cfg_dram_type[1]	1	O	14-2/14-5
LGPL2 /LOE	Local bus GP line 2 /output enable	LBC	cfg_core_pll[2]	1	O	14-2/14-5
LGPL3	Local bus GP line 3	LBC	cfg_boot_seq[0]	1	O	14-2/14-5
LGPL4 /LGTA/ LUPWAIT	Local bus GP line 4 /GPCM terminate access /UPM wait	LBC	—	1	I/O	14-2/14-5
LGPL5	Local bus GP line 5 address	LBC	cfg_boot_seq[1]	1	O	14-2/14-5

Table 3-2. Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	21-5/21-8
LSYNC_IN	Local bus DLL synchronization	LBC	—	1	I	14-2/14-5
LSYNC_OUT	Local bus DLL synchronization	LBC	—	1	O	14-2/14-5
$\overline{\text{LWE}}[1:3]/$ $\overline{\text{LBS}}[1:3]$	Local bus write enable /byte select 1–3	LBC	cfg_host_agt[0:2]	3	O	14-2/14-5
$\overline{\text{LWE0}}$ $\overline{\text{LBS0}}$	Local bus write enable /byte select 0	LBC	—	1	O	14-2/14-5
MA[15:0]	DDR address	DDR memory	—	16	O	9-3/9-5
MBA[2:0]	DDR bank select	DDR memory	—	3	O	9-3/9-5
$\overline{\text{MCAS}}$	DDR column address strobe	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MCK}}[0:5],$ $\overline{\text{MCK}}[0:5]$	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-3/9-5
MCKE[0:3]	DDR clock enable	DDR memory	—	4	O	9-3/9-5
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-5/10-8
$\overline{\text{MCS}}[0:3]$	DDR chip select (2/DIMM)	DDR memory	—	4	O	9-3/9-5
MDIC[0:1]	Driver impedance calibration	Debug/ DDR memory	—	2	I/O	
MDM[0:7]	DDR data mask	DDR memory	—	8	O	9-3/9-5
MDM8	DDR ECC data mask	DDR memory	—	1	O	9-3/9-5
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	9-3/9-5
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	9-3/9-5
$\overline{\text{MDQS}}[0:8]$	DDR ECC data strobe (complement)	DDR memory	—	9	I/O	9-3/9-5
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	9-3/9-5
MDVAL	Memory debug data valid	Debug	—	1	O	21-3/21-6
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
MODT[0:3]	DRAM On-Die Termination	DDR memory	—	4	O	
$\overline{\text{MRAS}}$	DDR row address strobe	DDR memory	—	1	O	9-3/9-5
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	21-3/21-6
MSRCID0	Memory debug source port ID 0	Debug	cfg_mem_debug	1	O	21-3/21-6
MSRCID1	Memory debug source port ID 1	Debug	cfg_ddr_debug	1	O	21-3/21-6
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	9-3/9-5
PCI_AD[31:0]	PCI address/data	PCI	—	32	I/O	17-2/17-6

Table 3-2. Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
PCI_C/ $\overline{\text{BE}}$ [3:0]	PCI command/byte enable	PCI	—	4	I/O	17-2/17-6
PCI_CLK	PCI clock	PCI	—	1	I	17-2/17-6
$\overline{\text{PCI\_DEVSEL}}$	PCI device select	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_FRAME}}$	PCI frame	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_GNT0}}$	PCI grant 0	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_GNT1}}$	PCI grant 1	PCI	cfg_pci_impd	1	O	17-2/17-6
$\overline{\text{PCI\_GNT2}}$	PCI grant 2	PCI	cfg_pci_arb	1	O	17-2/17-6
$\overline{\text{PCI\_GNT3}}$	PCI grant 3	PCI	cfg_pci_speed	1	O	17-2/17-6
$\overline{\text{PCI\_GNT4}}$	PCI grant 4	PCI	cfg_pci_clk	1	O	17-2/17-6
PCI_IDSEL	PCI initial device select	PCI	—	1	I	17-2/17-6
$\overline{\text{PCI\_IRDY}}$	PCI initiator ready	PCI	—	1	I/O	17-2/17-6
PCI_PAR	PCI parity	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_PERR}}$	PCI parity error	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_REQ}}$ [4:1]	PCI request 4–1	PCI	—	4	I	17-2/17-6
$\overline{\text{PCI\_REQ0}}$	PCI request 0	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_SERR}}$	PCI system error	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_STOP}}$	PCI stop	PCI	—	1	I/O	17-2/17-6
$\overline{\text{PCI\_TRDY}}$	PCI target ready	PCI	—	1	I/O	17-2/17-6
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
RTC	Real time clock	Clock	—	1	I	4-3/4-3
SD1_REF_CLK, $\overline{\text{SD1\_REF\_CLK}}$	SerDes1 reference clock, SerDes1 reference clock complement	PCI Express 1, PCI Express 2	—	2	I	18-2/18-5
SD1_RX[3:0], $\overline{\text{SD1\_RX}}$ [3:0]	Receive data, receive data complement	PCI Express 1	—	8	I	18-2/18-5
SD1_RX[7:4], $\overline{\text{SD1\_RX}}$ [7:4]	Receive data, receive data complement	PCI Express 2	—	8	I	18-2/18-5
SD1_TX[3:0], $\overline{\text{SD1\_TX}}$ [3:0]	Transmit data, transmit data complement	PCI Express 1	—	8	O	18-2/18-5
SD1_TX[7:4], $\overline{\text{SD1\_TX}}$ [7:4]	Transmit data, transmit data complement	PCI Express 2	—	8	O	18-2/18-5
SD2_REF_CLK, $\overline{\text{SD2\_REF\_CLK}}$	SerDes2 reference clock, SerDes2 reference clock complement	PCI Express 3	—	2	I	18-2/18-5
SD2_RX[0], $\overline{\text{SD2\_RX}}$ [0]	Receive data, receive data complement	PCI Express 3	—	4	I	18-2/18-5
SD2_TX[0], $\overline{\text{SD2\_TX}}$ [0]	Transmit data, transmit data complement	PCI Express 3	—	4	O	18-2/18-5

Table 3-2. Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
SYSCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
TCK	Test clock	JTAG	—	1	I	21-5/21-8
TDI	Test data in	JTAG	—	1	I	21-5/21-8
TDO	Test data out	JTAG	—	1	O	21-5/21-8
TEMP_ANODE	Thermal diode access	Test	—	2	I	21-5/21-8
TEMP_CATHODE	Thermal diode access	Test	—	2	I	21-5/21-8
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	21-5/21-8
TMS	Test mode select	JTAG	—	1	I	21-5/21-8
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	21-4/21-7
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	21-4/21-7
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	21-5/21-8
TSEC1_COL	TSEC1 collision detect	eTSEC1	FIFO1_TX_FC	1	I	15-2/15-9
TSEC1_CRS	TSEC1 carrier sense	eTSEC1	FIFO1_RX_FC	1	I/O	15-2/15-9
TSEC1_GTX_CLK	TSEC1 transmit clock out	eTSEC1	—	1	O	15-2/15-9
TSEC1_RX_CLK	TSEC1 receive clock	eTSEC1	FIFO1_RX_CLK	1	I	15-2/15-9
TSEC1_RX_DV	TSEC1 receive data valid	eTSEC1	FIFO1_RX_DV	1	I	15-2/15-9
TSEC1_RX_ER	TSEC1 receiver error	eTSEC1	FIFO1_RX_ER	1	I	15-2/15-9
TSEC1_RXD[7:0]	TSEC1 receive data	eTSEC1	FIFO1_RXD[7:0]	8	I	15-2/15-9
TSEC1_TX_CLK	TSEC1 transmit clock in	eTSEC1	FIFO1_TX_CLK	1	I	15-2/15-9
TSEC1_TX_EN	TSEC1 transmit enable	eTSEC1	FIFO1_TX_EN	1	O	15-2/15-9
TSEC1_TX_ER	TSEC1 transmit error	eTSEC1	FIFO1_TX_ER /cfg_etsec1_reduce	1	O	15-2/15-9
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	eTSEC1	FIFO1_TXD[6:4] /cfg_rom_loc[0:2]	3	O	15-2/15-9
TSEC1_TXD0	TSEC1 transmit data 0	eTSEC1	FIFO1_TXD0 /cfg_etsec1_prctl[0]	1	O	15-2/15-9
TSEC1_TXD1	TSEC1 transmit data 1	eTSEC1	FIFO1_TXD1 /cfg_etsec1_prctl[1]	1	O	15-2/15-9
TSEC1_TXD2	TSEC1 transmit data 2	eTSEC1	FIFO1_TXD2 /cfg_etsec1_serial	1	O	15-2/15-9
TSEC1_TXD3	TSEC1 transmit data 3	eTSEC1	FIFO1_TXD3	1	O	15-2/15-9
TSEC1_TXD7	TSEC1 transmit data 7	eTSEC1	FIFO1_TXD7	1	O	15-2/15-9
TSEC3_COL	TSEC3 collision detect	eTSEC3	FIFO3_TX_FC	1	I	15-2/15-9
TSEC3_CRS	TSEC3 carrier sense	eTSEC3	FIFO3_RX_FC	1	I/O	15-2/15-9
TSEC3_GTX_CLK	TSEC3 transmit clock out	eTSEC3	—	1	O	15-2/15-9
TSEC3_RX_CLK	TSEC3 receive clock	eTSEC3	FIFO3_RX_CLK	1	I	15-2/15-9

Table 3-2. Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC3_RX_DV	TSEC3 receive data valid	eTSEC3	FIFO3_RX_DV	1	I	15-2/15-9
TSEC3_RX_ER	TSEC3 receive error	eTSEC3	FIFO3_RX_ER	1	I	15-2/15-9
TSEC3_RXD[7:0]	TSEC3 receive data 7–0	eTSEC3	FIFO3_RXD[7:0]	8	I	15-2/15-9
TSEC3_TX_CLK	TSEC3 transmit clock in	eTSEC3	FIFO3_TX_CLK	1	I	15-2/15-9
TSEC3_TX_EN	TSEC3 transmit enable	eTSEC3	FIFO3_TX_EN	1	O	15-2/15-9
TSEC3_TX_ER	TSEC3 transmit error	eTSEC3	FIFO3_TX_ER cfg_etsec3_reduce	1	O	15-2/15-9
TSEC3_TXD[1:0]	TSEC3 transmit data 1–0	eTSEC3	FIFO3_TXD[1:0] cfg_etsec3_prctl[1:0]	2	O	15-2/15-9
TSEC3_TXD[6:4]	TSEC3 transmit data [6:4]	eTSEC3	FIFO3_TXD[6:4] /cfg_io_ports[0:2]	3	O	15-2/15-9
TSEC3_TXD2	TSEC3 transmit data 2	eTSEC3	FIFO3_TXD2 cfg_etsec3_serial	1	O	15-2/15-9
TSEC3_TXD3	TSEC3 transmit data 3	eTSEC3	FIFO3_TXD3	1	O	15-2/15-9
TSEC3_TXD7	TSEC3 transmit data 7	eTSEC3	FIFO3_TXD7	1	O	15-2/15-9
UART_CTS[0:1]	DUART clear to send	Dual UART	—	2	I	13-2/13-3
UART_RTS[0:1]	DUART ready to send	Dual UART	—	2	O	13-2/13-3
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	13-2/13-3
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	13-2/13-3
UDE	Unconditional debug event	PIC	—	1	I	10-5/10-8

## 3.2 Configuration Signals Sampled at Reset

The signals that serve alternate functions as configuration input signals during system reset are summarized in [Table 3-3](#). The detailed interpretation of their voltage levels during reset is described in [Chapter 4, “Reset, Clocking, and Initialization.”](#)

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of  $\overline{\text{HRESET}}$ . However, there is a setup and hold time for these signals relative to the rising edge of  $\overline{\text{HRESET}}$ , as described in the *MPC8533E Integrated Processor Hardware Specifications*. Note that the PLL configuration signals have different setup and hold time requirements than the other reset configuration signals.

The reset configuration signals are multiplexed with other functional signals. The values on these signals during reset are interpreted to be logic one or zero, regardless of whether the functional signal name is defined as active-low. Most of the reset configuration signals have internal pull-up resistors so that if the signals are not driven, the default value is high (a one), as shown in the table. Some signals do not have pull-up resistors and must be driven high or low during the reset period. For details about all the signals that require external pull-up resistors, see the *MPC8533E Integrated Processor Hardware Specifications*.

Note that the multiplexing of various signals on the MPC8533E is controlled by the PMUXCR register described in [Chapter 19, “Global Utilities.”](#)

**Table 3-3. MPC8533E Reset Configuration Signals**

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
PCI	$\overline{\text{PCI\_GNT4}}$	cfg_pci_clk	1
	$\overline{\text{PCI\_GNT3}}$	cfg_pci_speed	1
	$\overline{\text{PCI\_GNT2}}$	cfg_pci_arb	1
	$\overline{\text{PCI\_GNT1}}$	cfg_pci_impd	1
eTSEC1	TSEC1_TXD[6:4]	cfg_rom_loc[0:2]	111
	TSEC1_TXD[2]	cfg_etsec1_serial	1
	TSEC1_TXD[1:0]	cfg_etsec1_prtcl[1:0]	11
	TSEC1_TX_ER	cfg_tsec1_reduce	1
eTSEC3	TSEC3_TXD[6:4]	cfg_io_ports[0:2]	111
	TSEC3_TXD[2]	cfg_etsec3_serial	1
	TSEC3_TXD[1:0]	cfg_etsec3_prtcl[1:0]	11
	TSEC3_TX_ER	cfg_tsec3_reduce	1
LBC	$\overline{\text{LWE\_B}}[0]$	cfg_sec_freq	1
	LAD[0:31]	cfg_gpinput[0:31]	Indeterminate if not driven (no default)
	LA27	cfg_cpu_boot	1
	LA[28:31]	cfg_sys_pll[0:3]	Must be driven
	$\overline{\text{LWE}}[1:3]/\overline{\text{LBS}}[1:3]$	cfg_host_agt[0:2]	111
	LBCTL	cfg_core_pll[0]	Must be driven
	LALE	cfg_core_pll[1]	Must be driven
	LGPL2/ $\overline{\text{LOE}}$	cfg_core_pll[2]	Must be driven
	LGPL0	cfg_dram_type[0]	1
	LGPL1	cfg_dram_type[1]	1
	LGPL3	cfg_boot_seq[0]	1
	LGPL5	cfg_boot_seq[1]	1
	Debug	MSRCID0	cfg_mem_debug
MSRCID1		cfg_ddr_debug	1



### 3.3 Output Signal States During Reset

When a system reset is recognized ( $\overline{\text{HRESET}}$  is asserted), the MPC8533E aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See [Chapter 4, “Reset, Clocking, and Initialization,”](#) for a complete description of the reset functionality.

During reset, the MPC8533E ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state.

**Table 3-4. Output Signal States During System Reset**

Interface	Signal	State During Reset
DDR Memory	MBA[2:0]	High-Z
DDR Memory	MA[15:0]	High-Z
DDR Memory	$\overline{\text{MWE}}$	High-Z
DDR Memory	$\overline{\text{MRAS}}$	High-Z
DDR Memory	$\overline{\text{MCAS}}$	High-Z
DDR Memory	$\overline{\text{MCS}}[0:3]$	High-Z
DDR Memory	MCKE[0:3]	Driven <sup>1</sup>
DDR Memory	MCK[0:5], $\overline{\text{MCK}}[0:5]$	Driven Toggling
DDR Memory	MODT[0:3]	Driven Low
PCI Express	SD_TX[7:0], $\overline{\text{SD\_TX}}[7:0]$	High-Z
PCI Express	SD_PLL_TPD	High-Z
Analog	SD_PLL_TPA	Driven
TSEC1	TSEC1_TX_EN	High-Z
TSEC1	TSEC1_TX_ER	High-Z
TSEC1	TSEC1_GTX_CLK	High-Z
TSEC3	TSEC3_TX_EN	High-Z
TSEC3	TSEC3_GTX_CLK	High-Z
TSEC3	TSEC3_TXD[3]	Input—reset config (test only) <sup>2</sup>
TSEC3	TSEC3_TXD[6:7]	Input—reset config
TSEC3	TSEC3_TX_ER	Driven Low
LBC	LCLK[0:2]	High-Z
LBC	LCKE	High-Z
LBC	$\overline{\text{LCS}}[0:4]$	High-Z
LBC	$\overline{\text{DMA\_DACK2/LCS6}}$ $\overline{\text{DMA\_DDONE2/LCS7}}$	High-Z
DMA	$\overline{\text{DMA\_DACK}}[0:1]$	Input—reset config (test only) <sup>2</sup>
DMA	$\overline{\text{DMA\_DACK2/LCS6}}$ $\overline{\text{DMA\_DDONE2/LCS7}}$	High-Z

**Table 3-4. Output Signal States During System Reset (continued)**

Interface	Signal	State During Reset
DMA	$\overline{\text{DMA\_DDONE}}[0:1]$	High-Z
PIC	IRQ8	Driven (test only) <sup>2</sup>
PIC	$\overline{\text{IRQ\_OUT}}$	High-Z
Dual UART	UART_SOUT[0:1]	High-Z
Dual UART	$\overline{\text{UART\_RTS}}[0:1]$	High-Z
System Control	$\overline{\text{HRESET\_REQ}}$	Input—reset config (test only) <sup>2</sup>
System Control	$\overline{\text{CKSTP\_OUT}}$	High-Z
Debug	TRIG_OUT/READY	Input—reset config (test only) <sup>2</sup>
Debug	MSRCID[2:4]	Input—reset config (test only) <sup>2</sup>
Debug	MDVAL	High-Z
Power Mgmt	ASLEEP	Input—reset config (test only) <sup>2</sup>
Clock	CLK_OUT	Driven Toggling
GPOUT	GPOUT[0:7]	High-Z
JTAG	TDO	Driven

<sup>1</sup> Driven according to DRAM type. See [Section 4.4.3.9, “DDR SDRAM Type,”](#) on page 4-17 for specific voltage levels.

<sup>2</sup> Test-mode input during reset; see the *MPC8533E Integrated Processor Hardware Specifications* for details.

# Chapter 4

## Reset, Clocking, and Initialization

This chapter describes the reset, clocking, and some overall initialization of the MPC8533E, including a definition of the reset configuration signals and the options they select. Additionally, the configuration, control, and status registers are described. Note that other chapters in this book may describe specific aspects of initialization for individual blocks.

### 4.1 Overview

The reset, clocking, and control signals provide many options for the operation of the MPC8533E. Additionally, many modes are selected with reset configuration signals during a hard reset (assertion of  $\overline{\text{HRESET}}$ ).

### 4.2 External Signal Descriptions

Table 4-1 summarizes the external signals described in this chapter. Table 4-2 and Table 4-3 have detailed signal descriptions, but Table 4-1 contains references to additional sections that contain more information.

**Table 4-1. Signal Summary**

Signal	I/O	Description	References (Section/Page)
$\overline{\text{HRESET}}$	I	Hard reset input. Causes a power-on reset (POR) sequence.	<a href="#">4.4.1.2/4-8</a>
$\overline{\text{HRESET\_REQ}}$	O	Hard reset request output. An internal block requests that $\overline{\text{HRESET}}$ be asserted.	—
$\overline{\text{SRESET}}$	I	Soft reset input. Causes <i>mcp</i> assertion to the core	<a href="#">4.4.1.1/4-8</a>
READY	O	The MPC8533E has completed the reset operation and is not in a power-down (nap, doze or sleep) or debug state.	<a href="#">4.4.2/4-9</a>
SYSCLK	I	Primary clock input to the MPC8533E	<a href="#">4.4.4.1/4-21</a>
RTC	I	Real time clock input	<a href="#">4.4.4.4/4-23</a>
$\overline{\text{SD\_REF\_CLK}}$ / $\overline{\text{SD\_REF\_CLK}}$	I	SERDES high-speed interface reference clock	<a href="#">4.4.4.2/4-22</a>
$\overline{\text{SD2\_REF\_CLK}}$ / $\overline{\text{SD2\_REF\_CLK}}$	I	Second SERDES high-speed interface reference clock	<a href="#">4.4.4.2/4-22</a>

The following sections describe the reset and clock signals in detail.

## 4.2.1 System Control Signals

Table 4-2 describes some of the system control signals of the MPC8533E. Section 4.4.3, “Power-On Reset Configuration,” describes the signals that also function as reset configuration signals. Note that the  $\overline{\text{CKSTP\_IN}}$  and  $\overline{\text{CKSTP\_OUT}}$  signals are described in Chapter 19, “Global Utilities.”

**Table 4-2. System Control Signals—Detailed Signal Descriptions**

Signal	I/O	Description
$\overline{\text{HRESET}}$	I	Hard reset. Causes the MPC8533E to abort all current internal and external transactions and set all registers to their default values. $\overline{\text{HRESET}}$ may be asserted completely asynchronously with respect to all other signals.
		<b>State Meaning</b> Asserted/Negated—See Chapter 3, “Signal Descriptions,” and Section 4.4.3, “Power-On Reset Configuration,” for more information on the interpretation of the other MPC8533E signals during reset.
		<b>Timing</b> Assertion/Negation—The <i>MPC8533E Integrated Processor Hardware Specifications</i> gives specific timing information for this signal and the reset configuration signals.
$\overline{\text{HRESET\_REQ}}$	O	Hard reset request. Indicates to the board (system in which the MPC8533E is embedded) that a condition requiring the assertion of $\overline{\text{HRESET}}$ has been detected.
		<b>State Meaning</b> Asserted—A watchdog timer or a boot sequencer failure (see Section 11.4.5, “Boot Sequencer Mode”) has triggered a request for hard reset. Negated—Indicates no reset request.
		<b>Timing</b> Assertion/Negation—May occur any time, synchronous to the core complex bus clock. Once asserted, $\overline{\text{HRESET\_REQ}}$ does not negate until $\overline{\text{HRESET}}$ is asserted.
$\overline{\text{SRESET}}$	I	Soft reset. Causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts, the assertion of $\overline{\text{SRESET}}$ causes a core checkstop. $\overline{\text{SRESET}}$ need not be asserted during a hard reset.
		<b>State Meaning</b> Asserted—Asserting $\overline{\text{SRESET}}$ causes a machine check interrupt (edge sensitive) to the e500 core. $\overline{\text{SRESET}}$ has no effect while $\overline{\text{HRESET}}$ is asserted. However, the POR sequence is paused if $\overline{\text{SRESET}}$ is asserted during POR.
		<b>Timing</b> Assertion—May occur at any time, asynchronous to any clock. Negation—Must be asserted for at least two <i>CCB_clk</i> cycles.
READY	O	Ready. Multiplexed with TRIG_OUT and $\overline{\text{QUIESCE}}$ . See Chapter 21, “Debug Features and Watchpoint Facility,” for more information on TOSR and TRIG_OUT.
		<b>State Meaning</b> Asserted—Indicates that the MPC8533E has completed the reset operation and is not in a power-down state (nap, doze, or sleep) when TOSR[SEL] equals 0b000. See Section 4.4.2, “Power-On Reset Sequence,” for more information.
		<b>Timing</b> Assertion/Negation—Initial assertion of READY after reset is synchronous with SYSCLK. Subsequent assertion/negation due to power down modes occurs asynchronously.

## 4.2.2 Clock Signals

Table 4-3 describes the overall clock signals of the MPC8533E. Note that some clock signals are specific to blocks within the MPC8533E, and although some of their functionality is described in Section 4.4.4, “Clocking,” they are defined in detail in their respective chapters.

Note that there is also a CLK\_OUT signal in the MPC8533E; the signal driven on the CLK\_OUT pin is selectable and described in Section 19.4.1.23, “Clock Out Control Register (CLKOCR).”

**Table 4-3. Clock Signals—Detailed Signal Descriptions**

Signal	I/O	Description
SYSCLK	I	System clock/PCI clock (SYSCLK/PCI_CLK). SYSCLK is the primary clock input to the MPC8533E. It is the clock source for the e500 core and for all devices and interfaces that operate synchronously with the core. It is multiplied up with a phased-lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock), which is used by virtually all of the synchronous system logic, including the L2 cache, the DDR SDRAM and local bus memory controllers, and other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL in the e500 core and the PLL that creates the local bus memory clocks. When the PCI interface is used, SYSCLK also functions as the PCI_CLK signal. Note that this is true whether the MPC8533E is in agent or host mode. The MPC8533E does not provide a separate PCI_CLK output in host mode.
		<b>Timing</b> Assertion/Negation—See the <i>MPC8533E Integrated Processor Hardware Specifications</i> for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the e500 core. The RTC timing specifications are given in the <i>MPC8533E Integrated Processor Hardware Specifications</i> , but the maximum frequency should be less than one-quarter of the CCB frequency. See Section 4.4.4.4, “Real Time Clock.” This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).
		<b>Timing</b> Assertion/Negation—See the <i>MPC8533E Integrated Processor Hardware Specifications</i> for specific timing information for this signal.

## 4.3 Memory Map/Register Definition

This section describes the configuration and control registers that control access to the configuration space and to the boot code as well as guidelines for accessing these regions. It also contains a brief description of the boot sequencer which may be used to initialize configuration registers or memory before the CPU is released to boot.

### 4.3.1 Local Configuration Control

Table 4-4 shows the memory map for local configuration control registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 4-4. Local Configuration Control Register Map**

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-5
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-7

#### 4.3.1.1 Accessing Configuration, Control, and Status Registers

The configuration, control, and status registers are memory mapped. The set of configuration, control, and status registers occupies a 1-Mbyte region of memory. Their location is programmable using the CCSR base address register (CCSRBAR). The default base address for the configuration, control, and status registers is 0xFF70\_0000 (CCSRBAR = 0x000F\_F700). CCSRBAR itself is part of the local access block of CCSR memory, which begins at offset 0x0 from CCSRBAR. Because CCSRBAR is at offset 0x0 from the beginning of the local access registers, CCSRBAR always points to itself. The contents of CCSRBAR are broadcast internally in the MPC8533E to all functional units that need to be able to identify or create configuration transactions.

##### 4.3.1.1.1 Updating CCSRBAR

Updates to CCSRBAR that relocate the entire 1-Mbyte region of configuration, control, and status registers require special treatment. The effect of the update must be guaranteed to be visible by the

mapping logic before an access to the new location is seen. To make sure this happens, these guidelines should be followed:

- CCSRBAR should be updated during initial configuration of the device when only one host or controller has access to the device.
  - If the boot sequencer is being used to initialize, it is recommended that the boot sequencer set CCSRBAR to its desired final location.
  - If an external host on PCI is configuring the device, it should set CCSRBAR to the desired final location before the e500 core is released to boot.
  - If the e500 core is initializing the device, it should set CCSRBAR to the desired final location before enabling other I/O devices to access the device.
- When the e500 core is writing to CCSRBAR, it should use the following sequence:
  - Read the current value of CCSRBAR using a load word instruction followed by an **isync**. This forces all accesses to configuration space to complete.
  - Write the new value to CCSRBAR.
  - Perform a load of an address that does not access configuration space or the on-chip SRAM, but has an address mapping already in effect (for example, boot ROM). Follow this load with an **isync**.
  - Read the contents of CCSRBAR from its new location, followed by another **isync**.

#### 4.3.1.1.2 Configuration, Control, and Status Base Address Register (CCSRBAR)

Figure 4-1 shows the fields of CCSRBAR.

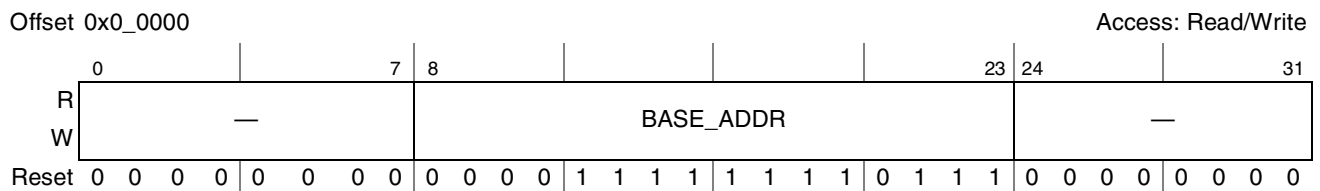


Figure 4-1. Configuration, Control, and Status Register Base Address Register (CCSRBAR)

Table 4-5 defines the bit fields of CCSRBAR.

Table 4-5. CCSRBAR Bit Settings

Bits	Name	Description
0–7	—	Write reserved, read = 0.
8–23	BASE_ADDR	Identifies the 16 most-significant address bits of the window used for configuration accesses. The base address is aligned on a 1-Mbyte boundary.
24–31	—	Write reserved, read = 0

### 4.3.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALTICAR registers. These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. By loading the proper boot sequencer command in the serial ROM, the base address in the ALTICBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 36-bit address that is mapped to the target specified in ALTICAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See [Section 11.4.5, “Boot Sequencer Mode,”](#) for more information.

#### NOTE

The enable bit in the ALTICAR register should be cleared either by the boot sequencer or by the boot code that executes after the boot sequencer has completed its configuration operations. This prevents problems with incorrect mappings if subsequent configuration of the local access windows uses a different target mapping for the address specified in ALTICBAR.

#### 4.3.1.2.1 Alternate Configuration Base Address Register (ALTICBAR)

Figure 4-2 shows the fields of ALTICBAR.

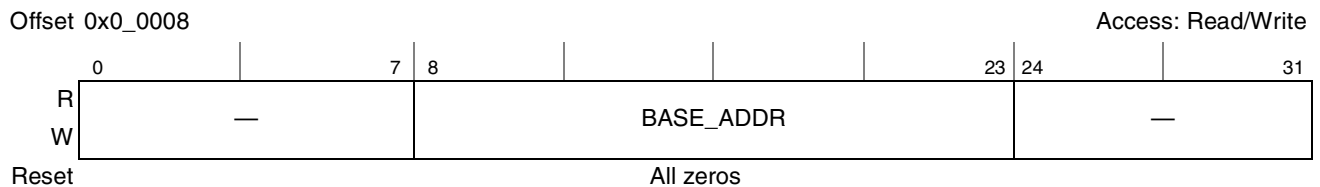


Figure 4-2. Alternate Configuration Base Address Register (ALTICBAR)

Table 4-6 defines the bit fields of ALTICBAR.

Table 4-6. ALTICBAR Bit Settings

Bits	Name	Description
0–7	—	Write reserved, read = 0
8–23	BASE_ADDR	Identifies the 16 most significant address bits of an alternate window used for configuration accesses.
24–31	—	Write reserved, read = 0

#### 4.3.1.2.2 Alternate Configuration Attribute Register (ALTICAR)

Figure 4-3 shows the fields of ALTICAR.

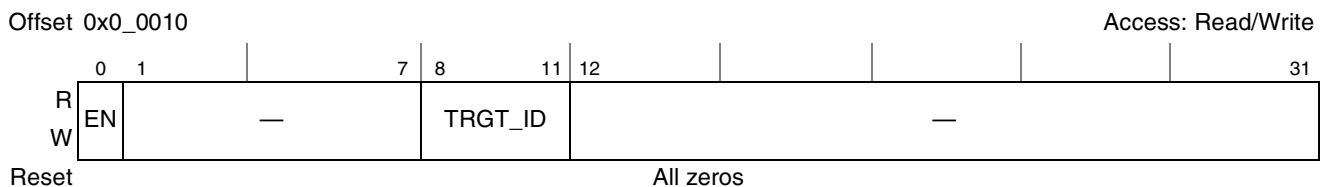


Figure 4-3. Alternate Configuration Attribute Register (ALTICAR)



Table 4-7 defines ALTCAR fields.

**Table 4-7. ALTCAR Bit Settings**

Bits	Name	Description														
0	EN	Enable for a second configuration window. Like CCSRBAR, it has a fixed size of 1 Mbyte. 0 Second configuration window is disabled. 1 Second configuration window is enabled.														
1–7	—	Write reserved, read = 0														
8–11	TRGT_ID	Identifies the device ID to target when a transaction hits in the 1-Mbyte address range defined by the second configuration window.  <table style="width: 100%; border: none;"> <tr> <td style="width: 50%;">0000 PCI Interface</td> <td style="width: 50%;">1001–1010 Reserved</td> </tr> <tr> <td>0001 PCI Express 2</td> <td>1011 Security</td> </tr> <tr> <td>0010 PCI Express 1</td> <td>1100 Reserved</td> </tr> <tr> <td>0011 PCI Express 3</td> <td>1101 Reserved</td> </tr> <tr> <td>0100 Local bus controller</td> <td>1110 Reserved</td> </tr> <tr> <td>0101–1011 Reserved</td> <td>1111 Local memory —DDR SDRAM and on-chip SRAM</td> </tr> <tr> <td>1000 Configuration, control, status registers</td> <td></td> </tr> </table>	0000 PCI Interface	1001–1010 Reserved	0001 PCI Express 2	1011 Security	0010 PCI Express 1	1100 Reserved	0011 PCI Express 3	1101 Reserved	0100 Local bus controller	1110 Reserved	0101–1011 Reserved	1111 Local memory —DDR SDRAM and on-chip SRAM	1000 Configuration, control, status registers	
0000 PCI Interface	1001–1010 Reserved															
0001 PCI Express 2	1011 Security															
0010 PCI Express 1	1100 Reserved															
0011 PCI Express 3	1101 Reserved															
0100 Local bus controller	1110 Reserved															
0101–1011 Reserved	1111 Local memory —DDR SDRAM and on-chip SRAM															
1000 Configuration, control, status registers																
12–31	—	Write reserved, read = 0														

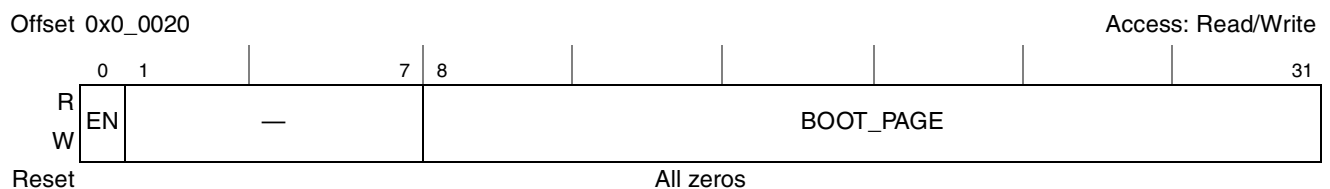
### 4.3.1.3 Boot Page Translation

When the e500 core comes out of reset, its MMU has one 4-Kbyte page defined at  $0x0\_FFFF\_Fnnn$ . The core begins execution with the instruction at effective address  $0x0\_FFFF\_FFFC$ . To get this instruction, the core's first instruction fetch is a burst read of boot code from effective address  $0x0\_FFFF\_FFE0$ . For systems in which the boot code resides at a different address, the MPC8533E provides boot page translation capability. Boot page translation is controlled by the boot page translation register (BPTR).

The boot sequencer can enable boot page translation, or the boot page translation can be set up by an external host when the MPC8533E is configured to be in boot holdoff mode. If translation is to be performed to a page outside the default boot ROM address range defined in the MPC8533E (8 Mbytes at  $0x0\_FF80\_0000$  to  $0x0\_FFFF\_FFFF$  as defined in [Section 4.4.3.4, “Boot ROM Location”](#)), the external host or boot sequencer must then also set up a local access window to define the routing of the boot code fetch to the target interface that contains the boot code, because the BPTR defines only the address translation, not the target interface. See [Section 2.1, “Local Memory Map Overview and Example,”](#) and [Section 11.4.5, “Boot Sequencer Mode,”](#) for more information.

#### 4.3.1.3.1 Boot Page Translation Register (BPTR)

Figure 4-4 shows the fields of BPTR.



**Figure 4-4. Boot Page Translation Register (BPTR)**

Table 4-8 describes BPTR bit settings.

**Table 4-8. BPTR Bit Settings**

Bits	Name	Description
0	EN	Boot page translation enable 0 Boot page is not translated. 1 Boot page is translated as defined in the BPTR[BOOT_PAGE] parameter.
1–7	—	Write reserved, read = 0
8–31	BOOT_PAGE	Translation for boot page. If enabled, the high order 24 bits of accesses to 0x0_FFFF_Fnnn are replaced with this value.

### 4.3.2 Boot Sequencer

The boot sequencer is a DMA engine that accesses a serial ROM on the I<sup>2</sup>C interface and writes data to CCSR memory or the memory space pointed to by the alternate configuration base address register (ALTCBAR). See [Section 4.3.1.2, “Accessing Alternate Configuration Space.”](#) The boot sequencer is enabled by reset configuration pins as described in [Section 4.4.3.8, “Boot Sequencer Configuration.”](#) If the boot sequencer is enabled, the e500 core is held in reset until the boot sequencer has completed its operation. For more details, see [Section 11.4.5, “Boot Sequencer Mode,”](#) in the I<sup>2</sup>C chapter.

## 4.4 Functional Description

This section describes the various ways to reset the MPC8533E, the POR configurations, and the clocking on the device.

### 4.4.1 Reset Operations

The MPC8533E has reset input signals for hard and soft reset operation.

#### 4.4.1.1 Soft Reset

Assertion of  $\overline{\text{SRESET}}$  causes a machine check interrupt to the e500 core. When this occurs, the soft reset flag is recorded in the machine check summary register (MCPSUMR) in the global utilities block so that software can identify the machine check as a soft reset condition. See the e500 core family reference manual for more information on the machine check interrupt and [Section 19.4.1.14, “Machine Check Summary Register \(MCPSUMR\),”](#) for more information on the setting of the soft reset flag. Note that if  $\overline{\text{SRESET}}$  is asserted before the e500 core is configured to handle a machine check interrupt, a core checkstop condition occurs, which causes  $\overline{\text{CKSTP\_OUT}}$  to assert.

#### 4.4.1.2 Hard Reset

The MPC8533E can be completely reset by the assertion of the  $\overline{\text{HRESET}}$  input. The assertion of this signal by external logic is the equivalent of a POR and causes the sequence of events described in [Section 4.4.2, “Power-On Reset Sequence.”](#)

Refer to the *MPC8533E Integrated Processor Hardware Specifications* for the timing requirements for  $\overline{\text{HRESET}}$  assertion and negation.

The hard reset request output signal ( $\overline{\text{HRESET\_REQ}}$ ) indicates to external logic that a hard reset is being requested by hardware or software. Hardware causes this signal to assert for a boot sequencer failure (see [Section 11.4.5, “Boot Sequencer Mode,”](#) and [Section 11.4.5.2, “EEPROM Data Format”](#)) or when the e500 watchdog timer is configured to cause a reset request when it expires. Software may request a hard reset by setting a bit in a global utilities register; see [Section 19.4.1.18, “Reset Control Register \(RSTCR\).”](#)

## 4.4.2 Power-On Reset Sequence

The POR sequence for the MPC8533E is as follows:

1. Power is applied to meet the specifications in the *MPC8533E Integrated Processor Hardware Specifications*.
2. The system asserts  $\overline{\text{HRESET}}$  and  $\overline{\text{TRST}}$ , causing all registers to be initialized to their default states and most I/O drivers to be three-stated (some clock, clock enabled, and system control signals are active).
3. The system applies a stable SYSCLK signal and stable PLL configuration inputs, and the device PLL begins locking to SYSCLK.
4. System negates  $\overline{\text{HRESET}}$  after its required hold time and after POR configuration inputs have been valid for at least 4 SYSCLK cycles.

### NOTE:

If the JTAG signals are not used, then  $\overline{\text{TRST}}$  may be tied negated. It is recommended that  $\overline{\text{TRST}}$  not remain asserted after the negation of  $\overline{\text{HRESET}}$ .  $\overline{\text{TRST}}$  may be connected directly to  $\overline{\text{HRESET}}$ .

There is no need to assert the  $\overline{\text{SRESET}}$  signal when  $\overline{\text{HRESET}}$  is asserted. If  $\overline{\text{SRESET}}$  remains asserted upon negation of  $\overline{\text{HRESET}}$ , the POR sequence will be paused after the e500 core PLL is locked and before the e500 reset is negated. The POR sequence will be resumed when  $\overline{\text{SRESET}}$  is negated.

5. MPC8533E enables I/O drivers.
6. The MPC8533E PCI interface can assert  $\overline{\text{DEVSEL}}$  in response to configuration cycles.
7. The e500 PLL configuration inputs are applied, allowing the e500 PLL to begin locking to the device clock (the CCB clock).
8. The CCB clock is cycled for approximately 50  $\mu\text{s}$  to lock the e500 PLL.
9. The internal hard reset to the e500 core is negated and soft resets are negated to the PLLs and other remaining I/O blocks. The PLLs begin to lock.
10. When PLL locking is completed, the boot sequencer is released, causing it to load configuration data from serial ROMs, if enabled, as described in [Section 4.4.3.8, “Boot Sequencer Configuration.”](#)

11. When the boot sequencer completes the PCI interface is released to accept external requests, and the boot vector fetch by the e500 core is allowed to proceed unless processor booting is further held off by POR configuration inputs as described in Section 4.4.3.7, “CPU Boot Configuration.” The MPC8533E is now in its ready state.
12. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state. The ready state is also indicated by the assertion of READY/TRIG\_OUT if TOSR[SEL] = 000. In this case, READY is asserted with the same rising edge of SYSCLK, to indicate that the device has reached its ready state. See Section 21.3.4.1, “Trigger Out Source Register (TOSR),” for more information on this register.

Asserting READY allows external system monitors to know basic device status, for example, exactly when it emerges from reset, or if the device is in a low-power mode. For more information on the debug functions of TRIG\_OUT, see Section 21.3.4, “Trigger Out Function.” For more information about power management states, see Section 19.4.1, “Register Descriptions.”

Figure 4-5 shows a timing diagram of the POR sequence.

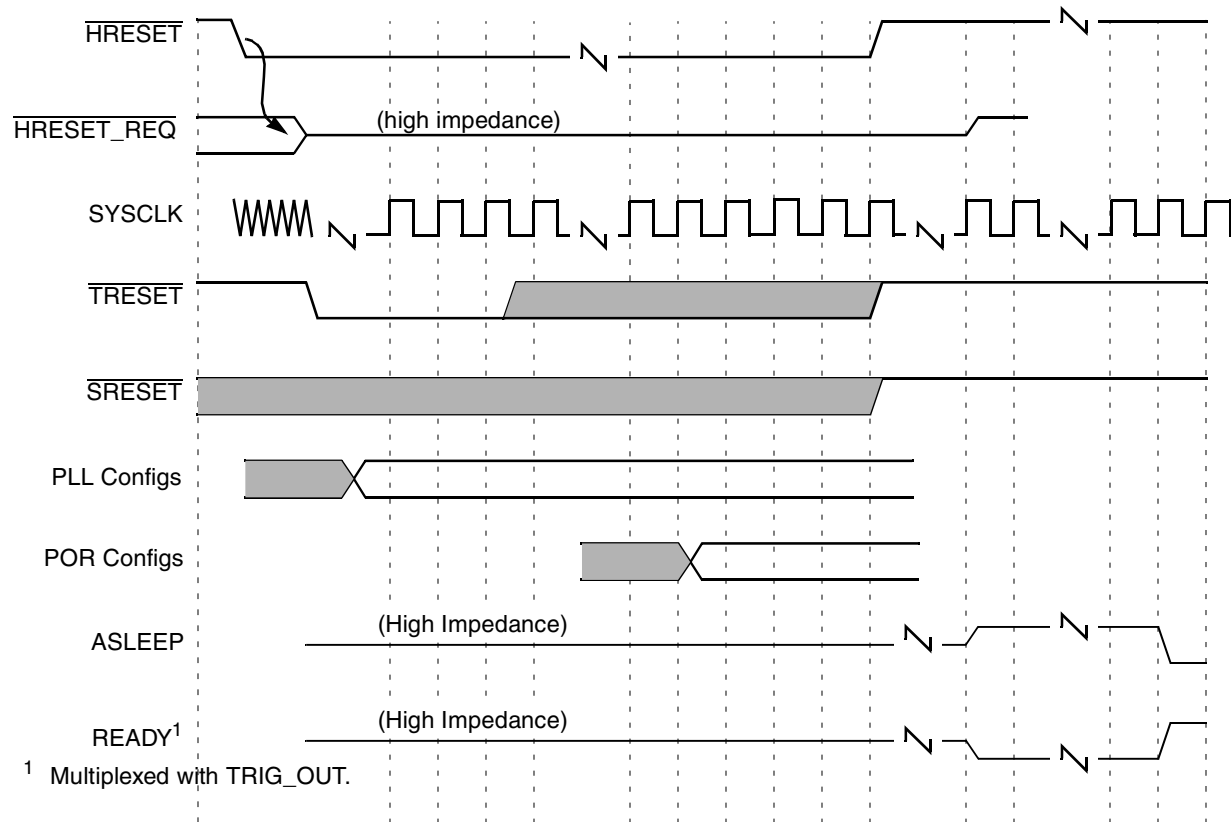


Figure 4-5. Power-On Reset Sequence

### 4.4.3 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during the assertion of  $\overline{\text{HRESET}}$ . The values of all these signals are sampled into registers while  $\overline{\text{HRESET}}$  is asserted. These inputs are to be pulled high or low by external resistors. During  $\overline{\text{HRESET}}$ , all other signal drivers connected to these signals must be in the high-impedance state.

Most POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board. Other POR configuration signals do not use pull-ups and therefore must be pulled high or low. Refer to the *MPC8533E Integrated Processor Hardware Specifications* for proper resistor values to be used for pulling POR configuration signals high or low.

This section describes the functions and modes configured by POR configuration signals. Note that many reset configuration settings are accessible to software through the following read-only memory-mapped registers described in [Chapter 19, “Global Utilities”](#):

- POR PLL status register (PORPLLSR)
- POR boot mode status register (PORBMSR)
- POR I/O impedance status and control register (PORIMPSCR)
- POR device status register (PORDEVSR)
- POR debug mode status register (PORDBGMSR)
- General-purpose POR configuration register (GPPORCR)—Reports the value on LAD[0:31] during POR (can be used to external system configuration)

#### NOTE

In the following tables, the binary value 0b0 represents a signal pulled down to GND and a value of 0b1 represents a signal pulled up to  $V_{DD}$ , regardless of the sense of the functional signal name on the signal.

### 4.4.3.1 System PLL Ratio

The system PLL inputs, shown in [Table 4-9](#), establish the clock ratio between the SYSCLK input and the platform clock used by the MPC8533E. The platform clock, also called the CCB clock, drives the L2 cache, the DDR SDRAM data rate, and the e500 core complex bus (CCB). There is no default value for this PLL ratio; these signals must be pulled to the desired values. See [Section 4.4.4.2.1, “Minimum Frequency Requirements,”](#) for optimal selection of this ratio with regard to available high-speed interface widths and frequencies. Note that the values latched on these signals during POR are accessible in the PORPLLSR (POR PLL status register), as described in [Section 19.4.1.1, “POR PLL Status Register \(PORPLLSR\).”](#)

Table 4-9. CCB Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (Binary)	CCB Clock : SYSCLK Ratio
LA[28:31]  No Default	cfg_sys_pll[0:3]	0000	16 : 1
		0001	Reserved
		0010	Reserved
		0011	3 : 1
		0100	4 : 1
		0101	5 : 1
		0110	6 : 1
		0111	Reserved
		1000	8 : 1
		1001	9 : 1
		1010	10 : 1
		1011	Reserved
		1100	12 : 1
		1101	Reserved
		1110	Reserved
1111	Reserved		

#### 4.4.3.2 e500 Core PLL Ratio

Table 4-10 describes the e500 core clock PLL inputs that program the core PLL and establish the ratio between the e500 core clock and the e500 core complex bus (CCB) clock. There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible through the memory-mapped PORPLLSR, as described in Section 19.4.1.1, “POR PLL Status Register (PORPLLSR),” and also in the e500 core HID1 register, as described in Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 4-10. e500 Core Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e500 Core: CCB ClockRatio
LBCTL, LALE, LGPL2/ $\overline{LOE}$ / $\overline{LSDRAS}$  No Default	cfg_core_pll[0:2]	000	4 : 1
		001	Reserved
		010	1 : 1
		011	3 : 2 (1.5 : 1)
		100	2 : 1
		101	5 : 2 (2.5:1)
		110	3 : 1
		111	7 : 2 (3.5 : 1)

### 4.4.3.3 SEC Frequency Ratio Configuration

The SEC mode configuration input, shown in [Table 4-11](#), is used to establish the ratio of the e500 Core Complex Bus (CCB) Clock to the SEC Clock. In the default 3:1 (CCB CLK: SEC CLK) setting, the SEC is clocked at one-third the frequency of the e500 Core Complex Bus (CCB) Clock. The ratio may be changed to 2:1 if the CCB Clock is below the threshold stated in the *MPC8533E Integrated Processor Hardware Specifications*.

**Table 4-11. SEC Mode Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LWE_B[0]	cfg_sec_freq	0	Enables the SEC in 2:1 (CCB CLK: SEC CLK) mode with respect to e500 Core Complex Bus (CCB) Clock
Default (1)		1	Enables the SEC in 3:1 (CCB CLK: SEC CLK) mode with respect to e500 Core Complex Bus (CCB) Clock (default).

Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR2 described in [Section 19.4.1.4](#), “POR Device Status Register (PORDEVSR).”

### 4.4.3.4 Boot ROM Location

The MPC8533E defines the default boot ROM address range to be 8 Mbytes at address 0x0\_FF80\_0000 to 0x0\_FFFF\_FFFF. However, which peripheral interface handles these boot ROM accesses can be selected at power on.

The boot ROM location inputs, shown in [Table 4-12](#), select the physical location of boot ROM. Accesses to the boot vector and the default boot ROM region of the local address map are directed to the interface specified by these inputs.

**Table 4-12. Boot ROM Location**

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[6:4]  Default (111)	cfg_rom_loc[0:2]	000	PCI
		001	DDR SDRAM
		010	PCI Express 2
		011	PCI Express 3
		100	PCI Express 1
		101	Local bus GPCM—8-bit ROM
		110	Local bus GPCM—16-bit ROM
		111	Local Bus GPCM—32-bit ROM (default)

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in [Section 19.4.1.2](#), “POR Boot Mode Status Register (PORBMSR).”

See [Section 2.1, “Local Memory Map Overview and Example,”](#) for an example memory map that relies on the default boot ROM values. Also, see [Section 4.3.1.3.1, “Boot Page Translation Register \(BPTR\),”](#) for information on translation of the boot page.

#### 4.4.3.5 Host/Agent Configuration

The host/agent reset configuration inputs, shown in [Table 4-13](#), configure the MPC8533E to act as a host or as an agent of a master on another interface. In host mode, the MPC8533E is immediately enabled to master transactions to the PCI interface. If the MPC8533E is an agent on the PCI or PCI Express interfaces, then the MPC8533E is disabled from mastering transactions on that interface until the external host enables it to do so. The external host does this by setting the control registers of the MPC8533E’s interfaces appropriately. See details in the PCI and PCI Express, programming models described in [Chapter 17, “PCI Bus Interface,”](#) and [Chapter 18, “PCI Express Interface Controller,”](#) respectively.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in [Section 19.4.1.2, “POR Boot Mode Status Register \(PORBMSR\).”](#)

**Table 4-13. Host/Agent Configuration**

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}[1:3]/\overline{\text{LBS}}[1:3]$	cfg_host_agt[0:2]	000	Reserved
Default (111)		001	MPC8533E acts as an endpoint on PCI-Express 3 interface. It acts as the host/root complex for all other PCI/PCI-Express interfaces.
		010	Reserved
		011	MPC8533E acts as an endpoint on PCI-Express 2 interface. It acts as the host/root complex for all other PCI/PCI-Express interfaces.
		100	Reserved
		101	MPC8533E acts as an endpoint on PCI-Express 1 interface. It acts as the host/root complex for all other PCI/PCI-Express interfaces.
		110	MPC8533E acts as an agent of an external host on its PCI interface. It acts as a root complex for all PCI-Express interfaces.
		111	MPC8533E acts as the host processor/root complex on all interfaces (default).

#### 4.4.3.6 I/O Port Selection

The MPC8533E can be configured with different I/O ports active. [Table 4-14](#) shows the configuration of I/O ports and bit rates (and required reference clocks) that are possible for the PCI Express interfaces.



Table 4-14. I/O Port Selection

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[6:4]  Default (111)	cfg_IO_ports[0:2]	000	All three PCI Express ports powered down
		001	Reserved
		010	PCI Express port 1 active PCI Express ports 2 and 3 powered down  PCI Express 1: RX lane[0:3] -> SD1_RX[0:3], TX lane[0:3] -> SD1_TX[0:3]
		011	Reserved
		100	PCI Express ports 1 and 2 active PCI Express port 3 powered down  PCI Express 1: RX lane[0:3] -> SD1_RX[0:3], TX lane[0:3] -> SD1_TX[0:3]  PCI Express 2: RX lane[0:3] -> SD1_RX[4:7], TX lane[0:3] -> SD1_TX[4:7]
		101	Reserved
	cfg_IO_ports[0:2]	110	All three PCI Express ports active  PCI Express 1: RX lane[0:3] -> SD1_RX[0:3], TX lane[0:3] -> SD1_TX[0:3]  PCI Express 2: RX lane[0:3] -> SD1_RX[4:7], TX lane[0:3] -> SD1_TX[4:7]  PCI Express 3: RX lane[0] -> SD2_RX[0], TX lane[0] -> SD2_TX[0]
		111	Reserved

#### 4.4.3.7 CPU Boot Configuration

The CPU boot configuration input, shown in [Table 4-15](#), specifies the boot configuration mode. If LA27 is sampled low at reset, the e500 core is prevented from fetching boot code until configuration by an external master is complete. The external master frees the CPU to boot by setting EEBPCR[CPU\_EN] in the ECM CCB port configuration register (EEBPCR). See [Section 8.2.1.2, “ECM CCB Port Configuration Register \(EEBPCR\),”](#) for more information.

Note that the value latched on this signal during POR is accessible through the memory-mapped PORBMSR (POR boot mode status register) described in [Section 19.4.1.2, “POR Boot Mode Status Register \(PORBMSR\).”](#)

Note also that the value latched on this signal during POR affects the PCI agent lock mode (See [Section 17.3.2.19, “PCI Bus Function Register \(PBFR\).”](#)) and the PCI Express Configuration Ready Register (See [Section 18.3.10.18, “Configuration Ready Register—0x4B0.”](#)).

**Table 4-15. CPU Boot Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LA27 Default (1)	cfg_cpu_boot	0	CPU boot holdoff mode. The e500 core is prevented from booting until configured by an external master.
		1	The e500 core is allowed to boot without waiting for configuration by an external master (default).

#### 4.4.3.8 Boot Sequencer Configuration

The boot sequencer configuration options, shown in [Table 4-16](#), allow the boot sequencer to load configuration data from the serial ROM located on the I<sup>2</sup>C1 port before the host tries to configure the MPC8533E. These options also specify normal or extended I<sup>2</sup>C addressing modes. See [Section 11.4.5, “Boot Sequencer Mode,”](#) for more information on the boot sequencer.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in [Section 19.4.1.2, “POR Boot Mode Status Register \(PORBMSR\).”](#)

**Table 4-16. Boot Sequencer Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LGPL3/LSDCAS, LGPL5 Default (11)	cfg_boot_seq[0:1]	00	Reserved
		01	Normal I <sup>2</sup> C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I <sup>2</sup> C interface. A valid ROM must be present.
		10	Extended I <sup>2</sup> C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I <sup>2</sup> C1 interface. A valid ROM must be present.
		11	Boot sequencer is disabled. No I <sup>2</sup> C ROM is accessed (default).

#### NOTE

When the boot sequencer is enabled, the processor core will be held in reset and thus prevented from fetching boot code until the boot sequencer has completed its task, regardless of the state of the CPU boot configuration signal described in [Section 4.4.3.7, “CPU Boot Configuration.”](#)

### 4.4.3.9 DDR SDRAM Type

DDR1 requires a different voltage level from DDR2. [Table 4-17](#) describes the configuration of the DDR SDRAM type.

**Table 4-17. DDR DRAM Type**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LGPL0, LGPL1 Default (11)	cfg_dram_type[0:1]	00	Reserved
		01	DDR1 2.5V, CKE low at reset
		10	Reserved
		11	DDR2 1.8V, CKE low at reset (default)

### 4.4.3.10 eTSEC1 Width

The eTSEC width input, shown in [Table 4-18](#), selects standard versus reduced width for three-speed Ethernet controller interface. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

This input does not affect the width of the FIFO interface which is always an 8-bit FIFO interface.

**Table 4-18. eTSEC1/eTSEC2 Width Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TX_ER Default (1)	cfg_tsec1_reduce	0	eTSEC1 interface operates in reduced pin mode, either RTBI, RGMII, RMII, or in 8-bit FIFO mode.
		1	eTSEC1 interface operates in standard width TBI, GMII, MII, or in 8-bit FIFO mode. (default)

### 4.4.3.11 eTSEC3 Width

The eTSEC3 width input, shown in [Table 4-19](#), selects standard versus reduced width for three-speed Ethernet controller interface 3. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

The value of this configuration setting does not affect the width of the FIFO interface on eTSEC3, which is always 8 bits.

Table 4-19. eTSEC3/eTSEC4 Width Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TX_ER  Default (1)	cfg_tsec3_reduce	0	eTSEC3 Ethernet interface operates in reduced mode, either RTBI, RGMII or RMII, or 8-bit FIFO mode.
		1	eTSEC3 Ethernet interface operates in standard TBI, GMII, MII, or 8-bit FIFO mode. (default).

#### 4.4.3.12 eTSEC1 Protocol

The eTSEC1 protocol inputs, shown in [Table 4-20](#), select the protocol (FIFO, MII, GMII or TBI) used by the eTSEC1 controller. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

Table 4-20. eTSEC1 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[0:1]  Default (11)	cfg_tsec1_prctl[0:1]	00	The eTSEC1 controller operates using 8-bit FIFO protocol.
		01	The eTSEC1 controller operates using the MII protocol (or RMII if configured in reduced mode as described in <a href="#">Section 4.4.3.10, “eTSEC1 Width”</a> ).
		10	The eTSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in <a href="#">Section 4.4.3.10, “eTSEC1 Width”</a> ).
		11	The eTSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in <a href="#">Section 4.4.3.10, “eTSEC1 Width”</a> ) (default).

#### 4.4.3.13 eTSEC3 Protocol

The eTSEC3 protocol inputs, shown in [Table 4-21](#), select the protocol (FIFO, MII, GMII or TBI) used by the eTSEC3 controller. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

Table 4-21. eTSEC3 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[0:1]  Default (11)	cfg_tsec3_prtcl[0:1]	00	The eTSEC3 controller operates using 8-bit FIFO protocol.
		01	The eTSEC3 controller operates using the MII protocol (or RMII if configured in reduced mode as described in <a href="#">Section 4.4.3.11, “eTSEC3 Width”</a> )
		10	The eTSEC3 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in <a href="#">Section 4.4.3.11, “eTSEC3 Width”</a> ).
		11	The eTSEC3 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in <a href="#">Section 4.4.3.11, “eTSEC3 Width”</a> ) (default).

#### 4.4.3.14 PCI Clock Selection

The PCI clock source inputs, shown in [Table 4-22](#) specify the clock mode (synchronous or asynchronous) for the PCI interface. See [Section 4.4.4.1, “System Clock/PCI Clock”](#) for more information. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in [Section 18.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

Table 4-22. PCI Clock Select

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
PCI_GNT[4]  Default (1)	cfg_pci_clk	0	Asynchronous mode. PCI_CLK is used as the clock for the PCI interface
		1	Synchronous mode. SYSCLK is used as the clock for the PCI interface. (default)

#### 4.4.3.15 PCI Speed Configuration

The PCI speed configuration input, shown in [Table 4-23](#), configures internal logic for proper operation with the PCI clock frequencies in use. The default setting is appropriate for PCI operating above 33 MHz. For low speed operation (PCI at or below 33 MHz) this POR configuration input should be low during  $\overline{\text{HRESET}}$ . If this configuration is not set properly, behavior of the PCI interface may be unreliable. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR, described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

Table 4-23. PCI Speed Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
PCI_GNT[3]  Default (1)	cfg_pci_speed	0	PCI frequency at or below 33 MHz
		1	PCI frequency above 33 MHz (default)

#### 4.4.3.16 PCI I/O Impedance

The PCI I/O impedance configuration inputs, shown in [Table 4-24](#) select the impedance of the PCI I/O drivers for the respective interfaces. Note that the values latched on these signals during POR are accessible through PORIMPSCR, described in [Section 19.4.1.3, “POR I/O Impedance Status and Control Register \(PORIMPSCR\).”](#)

**Table 4-24. PCI I/O Impedance**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
PCI_GNT1 Default (1)	cfg_pci_impd	0	25-Ω I/O drivers are used on the PCI interface.
		1	42-Ω I/O drivers are used on the PCI interface (default).

#### 4.4.3.17 PCI Arbiter Configuration

The PCI arbiter configuration inputs, shown in [Table 4-25](#), enable the on-chip PCI arbiter. Note that the value latched on these signals during POR are accessible through the PORDEVSR described in [Section 19.4.1.4, “POR Device Status Register \(PORDEVSR\).”](#)

**Table 4-25. PCI Arbiter Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
PCI_GNT2 Default (1)	cfg_pci_arb	0	The on-chip PCI arbiter is disabled. External arbitration is required.
		1	The on-chip PCI arbiter is enabled (default).

#### 4.4.3.18 Memory Debug Configuration

The memory debug configuration input, shown in [Table 4-26](#), selects which debug outputs (DDR or LBC memory controller) are driven onto the MSRCID and MDVAL debug signals. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in [Section 19.4.1.5, “POR Debug Mode Status Register \(PORDBGMSR\).”](#)

**Table 4-26. Memory Debug Configuration**

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
MSRCID0 Default (1)	cfg_mem_debug	0	Debug information from the local bus controller (LBC) is driven on the MSRCID and MDVAL signals
		1	Debug information from the DDR SDRAM controller is driven on the MSRCID and MDVAL signals (default).

#### 4.4.3.19 DDR Debug Configuration

The DDR debug configuration input, shown in [Table 4-27](#), enables a DDR memory controller debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto the ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8533E in this mode.

Table 4-27. DDR Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
MSRCID1 Default (1)	cfg_ddr_debug	0	Debug information is driven on the ECC pins instead of normal ECC I/O. ECC signals from memory devices must be disconnected.
		1	Debug information is not driven on ECC pins. ECC pins function in their normal mode (default).

Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in [Section 19.4.1.5, “POR Debug Mode Status Register \(PORDBGMSR\).”](#)

#### 4.4.3.20 General-Purpose POR Configuration

The LBC address/data bus inputs, shown in [Table 4-28](#), configure the value of the general-purpose POR configuration register defined in [Section 19.4.1.7, “General-Purpose POR Configuration Register \(GPPORCR\).”](#) This register is intended to facilitate POR configuration of user systems. A value placed on LAD[0:31] during POR is captured and stored (read only) in the GPPORCR. Software can then use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

Table 4-28. General-Purpose POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LAD[0:31] No default	cfg_gpinput[0:31]	—	General-purpose POR configuration vector to be placed in GPPORCR

### 4.4.4 Clocking

The following paragraphs describe the clocking within the MPC8533E device.

#### 4.4.4.1 System Clock/PCI Clock

The MPC8533E takes a single input clock, SYSCLK, as its primary clock source for the e500 core and all of the devices and interfaces that operate synchronously with the core. As shown in [Figure 4-6](#), the SYSCLK input (frequency) is multiplied up using a phase lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock). The CCB clock is used by virtually all of the synchronous system logic, including the L2 cache, and other internal blocks such as the DMA and interrupt controller. The CCB clock also feeds the PLL in the e500 core and the PLL that create clocks for the local bus memory controller. Note that the divide-by-two CCB clock divider and the divide-by-*n* CCB clock divider, shown in [Figure 4-6](#), are located in the DDR and local bus blocks, respectively.

The PCI interface may use SYSCLK as the PCI clock and thus have PCI operation be synchronous with the platform. Alternately, a separate, independent clock may be used for the PCI interface, in which case PCI operation is asynchronous with respect to SYSCLK and the platform clock.

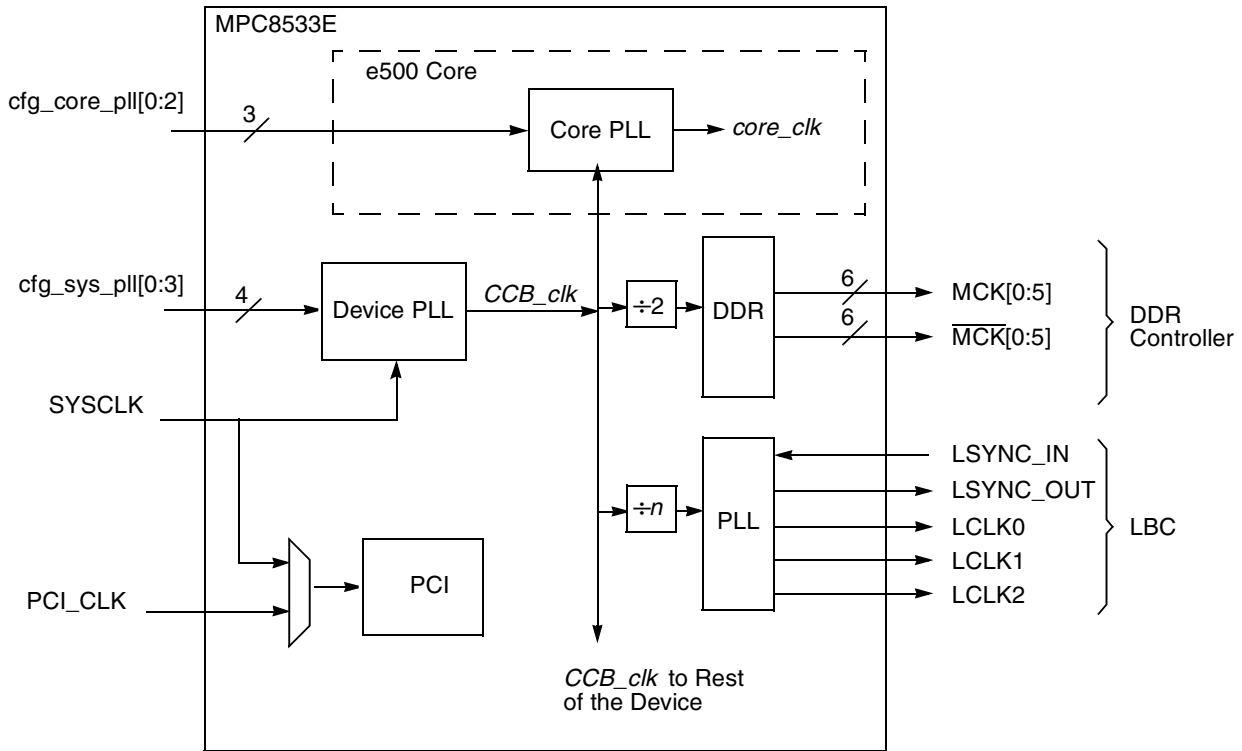


Figure 4-6. Clock Subsystem Block Diagram

#### 4.4.4.2 PCI Express

Clocks for these high speed interfaces on the MPC8533E are derived from a PLL in the SerDes block. This PLL is driven by a reference clock (SD\_REF\_CLK/SD\_REF\_CLK) whose input frequency is a function of the protocol and bit rate being used as shown in [Table 4-29](#).

Table 4-29. High Speed Interface Clocking

Interfaces	Bit Rate	Reference Clock Frequency
PCI Express	2.5 Gbps	100 MHz (Spread Spectrum supported)

##### 4.4.4.2.1 Minimum Frequency Requirements

[Section 4.4.3.6, “I/O Port Selection,”](#) describes various high-speed interface configuration options. Note that the CCB clock frequency must be considered for proper operation of such interfaces as described below.

For proper PCI Express operation, the CCB clock frequency must be greater than:

$$\frac{500 \text{ MHz} \times (\text{PCI Express link width})}{8}$$

8



See [Section 18.1.3.2, “Link Width,”](#) for PCI Express interface width details.

Note that the minimum CCB:SYSCLK ratio for PCI in synchronous mode is 6:1. See [Section 4.4.3.1, “System PLL Ratio,”](#) for details of selecting this ratio.

#### 4.4.4.3 Ethernet Clocks

The Ethernet blocks operate asynchronously with respect to the rest of the device. These blocks use receive and transmit clocks supplied by their respective PHY chips, plus a 125-MHz clock input for gigabit protocols. Data transfers are synchronized to the CCB clock internally.

#### 4.4.4.4 Real Time Clock

As shown in [Figure 4-7](#), the real time clock (RTC) input can optionally be used to clock the e500 core timer facilities. RTC can also be used (optionally) by the MPC8533E programmable interrupt controller (PIC) global timer facilities. The RTC is separate from the e500 core clock and is intended to support relatively low frequency timing applications. The RTC frequency range is specified in the *MPC8533E Integrated Processor Hardware Specifications*, but the maximum value should not exceed one-quarter of the CCB Frequency.

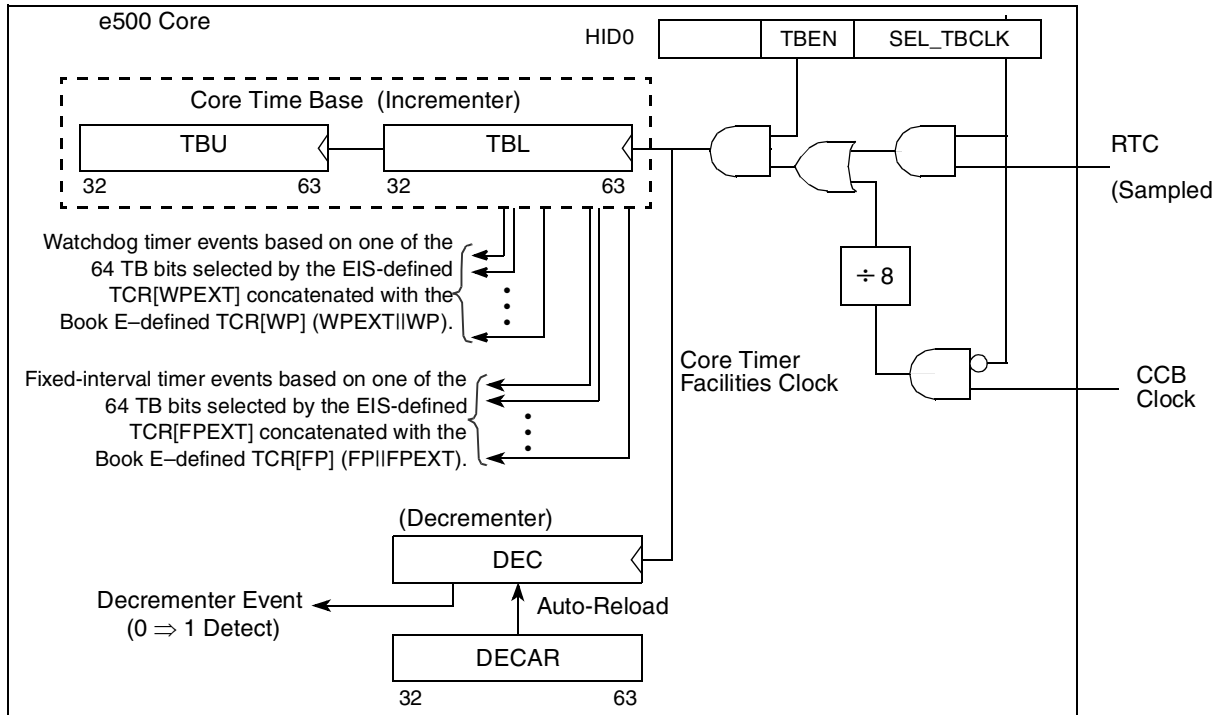
Before being distributed to the core time base, RTC is sampled and synchronized with the CCB clock.

The clock source for the core time base is specified by two fields in HID0: time base enable (TBEN), and select time base clock (SEL\_TBCLK). If the time base is enabled, (HID0[TBEN] is set), the clock source is determined as follows:

- HID0[SEL\_TBCLK] = 0, the time base is updated every 8 CCB clocks
- HID0[SEL\_TBCLK] = 1, the time base is updated on the rising edge of RTC

The default source of the time base is the CCB clock divided by eight. For more details, see the e500 core family reference manual.

[Section 10.3.2.6, “Timer Control Register \(TCR\),”](#) provides additional information on the use of the RTC signal to clock the global timers in the PIC unit.



**Note:** The logic circuits shown depict functional relationships only; they do not represent physical implementation details.

**Figure 4-7. RTC and Core Timer Facilities Clocking Options**

## Part II

# e500 Core Complex and L2 Cache

This part describes the many features of the MPC8533E core processor at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- [Chapter 5, “Core Complex Overview,”](#) provides an overview of the e500v2 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- [Chapter 6, “Core Register Summary,”](#) provides a listing of the e500v2 registers in reference form.
- [Chapter 7, “L2 Look-Aside Cache/SRAM,”](#) describes the L2 cache of the MPC8533E. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the Power ISA definition. This part provides additional information about the architecture as it relates specifically to the e500 core complex and specific details on how its registers are accessed.

The e500 core complex interacts with the L2 cache through the core complex bus (CCB).



## Chapter 5

# Core Complex Overview

This chapter provides an overview of the e500 microprocessor core as it is implemented on the MPC8533E.

References to e500 are true for both the e500v1 and e500v2.

This chapter includes the following:

- An overview of architecture features as implemented in this core and a summary of the core feature set
- A summary of the instruction pipeline and flow
- An overview of the programming model
- An overview of interrupts and exception handling
- A description of the memory management architecture
- High-level details of the e500 core memory and coherency model
- A brief description of the core complex bus (CCB)
- A summary of the Power Architecture embedded category compatibility and migration from the original version of the PowerPC™ architecture as it is defined by Apple, IBM, and Motorola (referred to as the AIM version of the PowerPC architecture)

Specific details about the e500 are provided in the *PowerPC™ e500 Core Family Reference Manual* (Freescale Document ID No. E500CORERM). The e500 core provides features that the integrated device may not implement or may implement in a more specific way.

## 5.1 Overview

The e500 core is a low-power implementation of the resources for embedded processors defined by the Power ISA. The core is a 32-bit implementation using the lower words in the 64-bit general-purpose registers (GPRs).

[Figure 5-1](#) is a block diagram of the processor core complex that shows how the functional units operate independently and in parallel. Note that this conceptual diagram does not attempt to show how these features are physically implemented.

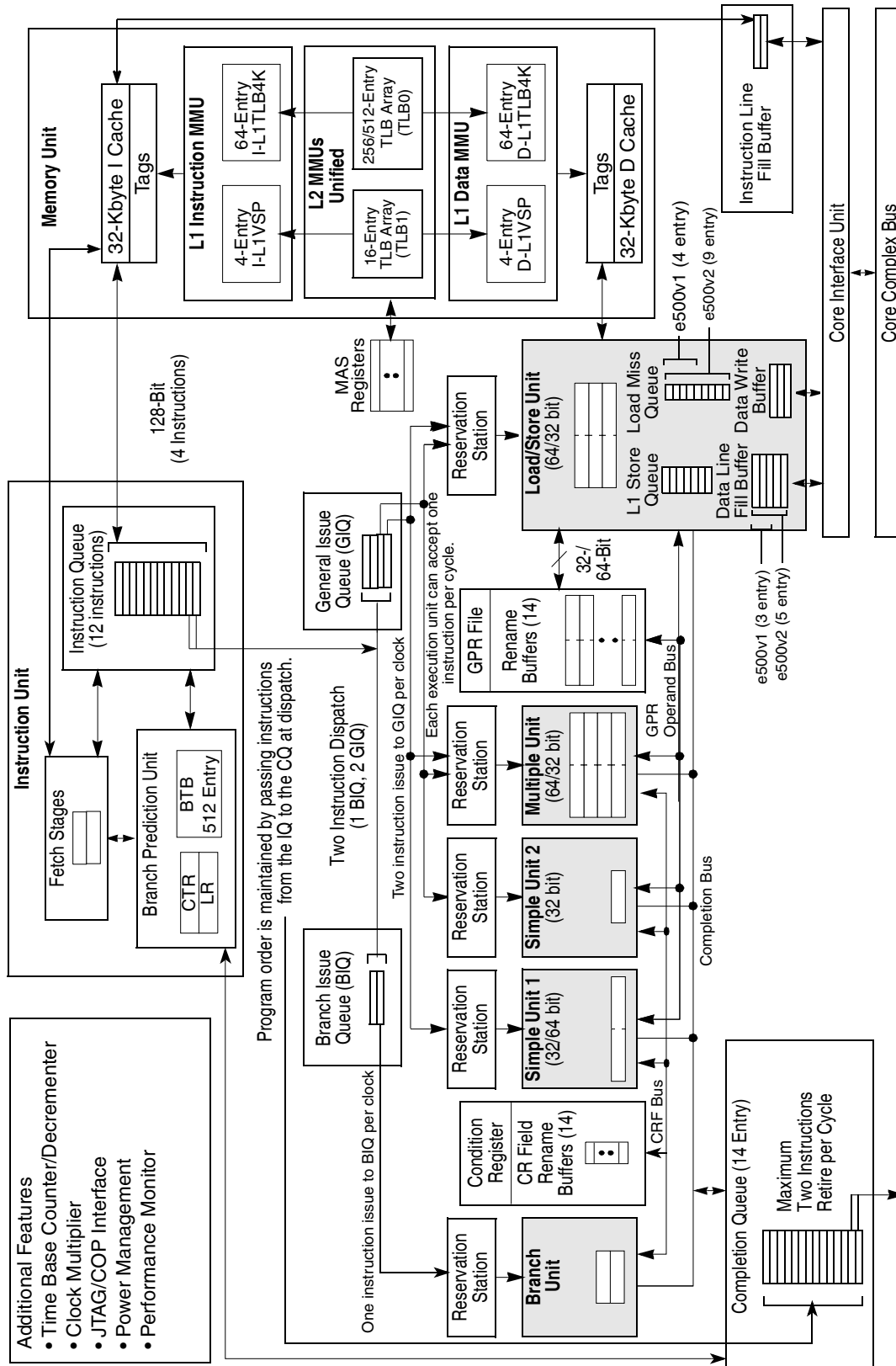


Figure 5-1. e500 Core Complex Block Diagram

The Power Architecture technology defines categories that extend the architecture that can perform computational or system management functions. One of these on the e500 is the signal processing engine (SPE), which includes a suite of vector instructions that use the upper and lower halves of the GPRs as a single two-element operand. Some extensions are defined by Freescale's embedded category implementation standards (EIS).

### 5.1.1 Upward Compatibility

The e500 provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits, as defined by the architecture. It also provides two-element, 64-bit data types for the SPE and embedded vector floating-point instructions, which include instructions that operate on operands comprised of two 32-bit elements. It also provides a 64-bit scalar data type for use with embedded double-precision floating-point instructions.

The embedded single-precision scalar floating-point instructions use 32-bit single-precision instructions.

#### NOTE

The SPE (which includes embedded floating-point functionality) is implemented in all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or embedded floating-point instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale Semiconductor offers a `libcfs1_e500` library that uses SPE instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

### 5.1.2 Core Complex Summary

The core complex is a superscalar processor that can issue two instructions and complete two instructions per clock cycle. Instructions complete in order, but can execute out of order. Execution results are available to subsequent instructions through the rename buffers, but those results are recorded into architected registers in program order, maintaining a precise exception model. All arithmetic instructions that execute in the core operate on data in the GPRs. Although the GPRs are 64 bits wide, only SPE, DPFP (e500v2 only), and embedded vector floating-point instructions operate on the upper word of the GPRs; the upper 32 bits are not affected by other 32-bit instructions.

The processor core integrates two simple instruction units (SU1, SU2), a multiple-cycle instruction unit (MU), a branch unit (BU), and a load/store unit (LSU).

The LSU and SU2 support 64- and 32-bit instructions.

The ability to execute five instructions in parallel and the use of simple instructions with short execution times yield high efficiency and throughput. Most integer instructions execute in 1 clock cycle. A series of

independent vector floating-point add instructions can be issued and completed with a throughput of one instruction per cycle.

The core complex includes independent on-chip, 32-Kbyte, eight-way set-associative, physically addressed caches for instructions and data. It also includes on-chip first-level instruction and data memory management units (MMUs) and an on-chip second-level unified MMU.

- The first-level MMUs contain two four-entry, fully-associative instruction and data translation lookaside buffer (TLB) arrays that provide support for demand-paged virtual memory address translation and variable-sized pages. They also contain two 64-entry, 4-way set-associative instruction and data TLB arrays that support 4-Kbyte pages. These arrays are maintained entirely by the hardware with a true least-recently-used (LRU) algorithm.
- The second-level MMU contains a 16-entry, fully-associative unified (instruction and data) TLB array that provides support for variable-sized pages. It also contains a unified TLB for 4-Kbyte page size support, as follows:
  - a 256-entry, 2-way set-associative unified TLB for the e500v1
  - a 512-entry, 4-way set-associative unified TLB for the e500v2

These second-level TLBs are maintained completely by the software.

The core complex allows cache-line-based user-mode locks on the contents in either the instruction or data cache. This provides embedded applications with the capability for locking interrupt routines or other important (time-sensitive) instruction sequences into the instruction cache. It also allows data to be locked into the data cache, which supports deterministic execution time.

The core complex supports a high-speed on-chip internal bus with data tagging called the core complex bus (CCB). The CCB has two general-purpose read data buses, one write data bus, data parity bits, data tag bits, an address bus, and address attribute bits. The processor core complex supports out-of-order reads, in-order writes, and one level of pipelining for addresses with address-retry responses. It can also support single-beat and burst data transfers for memory accesses and memory-mapped I/O operations.

## 5.2 e500 Processor and System Version Numbers

Table 5-1 lists the revision codes in the processor version register (PVR) and the system version register (SVR). These registers can be accessed as SPRs through the e500 core (see Section 6.5.3, “Processor Version Register (PVR),” and Section 6.5.4, “System Version Register (SVR)”) or as memory-mapped registers defined by the integrated device (see “Section 19.4.1.16, “Processor Version Register (PVR),” and Section 19.4.1.17, “System Version Register (SVR)”).

**Table 5-1. Device Revision Level Cross-Reference**

MPC8533E Revision	Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.0	2.1	0x8021_0021	0x803C_0010 for MPC8533E (with security) 0x8034_0010 for MPC8533 (without security)
1.1	2.2	0x8021_0022	0x803C_0011 for MPC8533E (with security) 0x8034_0011 for MPC8533 (without security)



## 5.3 Features

Key features of the e500 are summarized as follows:

- 32-bit architecture
- Additional categories (formerly referred to as APUs)

Branch target buffer (BTB) locking is specific to the e500. BTB locking gives the user the ability to lock, unlock, and invalidate BTB entries; further information is provided in [Table 5-5](#). The EIS (see *EREF: a Reference for Freescale Book E and the e500 Core*) defines the following:

- Integer select. This instruction is now part of the Power Architecture technology base category.
- Performance monitor. The performance monitor facility provides the ability to monitor and count predefined events such as processor clocks, misses in the instruction cache or data cache, types of instructions decoded, or mispredicted branches. The count of such events can be used to trigger the performance monitor exception. Additional performance monitor registers (PMRs) similar to SPRs are used to configure and track performance monitor operations. These registers are accessed with the Move to PMR and Move from PMR instructions (**mtpmr** and **mfpmr**). See [Section 5.12, “Performance Monitoring.”](#)
- Cache locking. Allows instructions and data to be locked into their respective caches on a cache block basis. Locking is performed by a set of touch and lock set instructions. This functionality can be enabled for user mode by setting MSR[UCLE]. The feature also provides resources for detecting and handling overlocking conditions.
- Machine check. The machine check interrupt is treated as a separate level of interrupt. It uses its own save and restore registers (MCSR0 and MCSR1) and Return from Machine Check Interrupt (**rfmci**) instruction. See [Section 5.8, “Interrupts and Exception Handling.”](#)
- Single-precision embedded scalar and vector floating-point instructions, listed in [Table 5-4](#).
- Signal processing engine (SPE). Note that the SPE is not a separate unit; SPE computational and logical instructions are executed in the simple and multiple-cycle units used by all other computational and logical instructions, and 64-bit loads and stores are executed in the common LSU. [Figure 5-1](#) shows how execution logic for SU1, the MU, and the LSU is replicated to support operations on the upper halves of the GPRs.

### NOTE

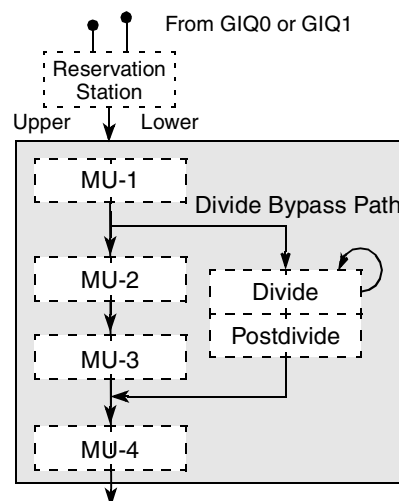
The SPE APU and the two single-precision floating-point APUs were combined in the original implementation of the e500 v1, as shown in [Figure 5-2](#).

Vector and Floating-Point APUs		e500 v1	e500 v2
Original SPE Definition	SPE vector instructions <b>ev...</b>	√	√
	Vector single-precision floating-point <b>evfs...</b>	√	√
	Scalar single-precision floating-point <b>efs...</b>	√	√
	Scalar double-precision floating-point <b>efd...</b>		√

**Figure 5-2. Vector and Floating-Point APUs**

- The e500 register set is modified as follows:
  - GPRs are widened to 64 bits to support 64-bit load, store, and merge operations. Note that the upper 32 bits are affected only by 64-bit instructions.
  - A 64-bit accumulator (ACC) has been added.
  - The signal processing and embedded floating-point status and control register (SPEFSCR) provides interrupt control and status for SPE and embedded floating-point instructions. These registers are shown in [Figure 5-7](#). SPE instructions are grouped as follows:
    - Single-cycle integer add and subtract with the same latencies for SPE operations as for the 32-bit equivalent
    - Single-cycle logical operations
    - Single-cycle shift and rotates
    - Four-cycle integer pipelined multiplies
    - 4-, 11-, 19-, and 35-cycle integer divides
    - If **rA** or **rB** is zero, a floating-point divide takes 4 cycles; all other cases take 29 cycles.
    - Four-cycle SIMD pipelined multiply-accumulate (MAC)
    - 64-bit accumulator for no-stall MAC operations
    - 64-bit loads and stores
    - 64-bit merge instructions
- Cache structure—Separate 32-Kbyte, 32-byte line, 8-way set-associative level 1 instruction and data caches
  - 1.5-cycle cache array access, 3-cycle load-to-use latency
  - Pseudo-LRU (PLRU) replacement algorithm
  - Copy-back data cache that can function as a write-through cache on a page-by-page basis
  - Supports all embedded category memory coherency modes
  - Supports EIS-defined cache-locking instructions, as listed in [Table 5-3](#)
- Dual-issue superscalar control
  - Two-instructions-per-clock peak issue rate
  - Precise exception handling
- Decode unit
  - 12-entry instruction queue (IQ)
  - Full hardware detection of interlocks
  - Decodes as many as two instructions per cycle
  - Decode serialization control
  - Register dependency resolution and renaming
- Branch prediction unit (BPU)
  - Dynamic branch prediction using a 512-entry, 4-way set-associative branch target buffer (BTB) supported by the e500 BTB instructions listed in [Table 5-5](#).
  - Branch prediction is handled in the fetch stages.

- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue (CQ)
  - In-order retirement of as many as two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts, mispredicted branches, and context-synchronizing instructions
- Issue queues
  - Two-entry branch instruction issue queue (BIQ)
  - Four-entry general instruction issue queue (GIQ)
- Branch unit—The branch unit (BU) is an execution unit and is distinct from the BPU. It executes (resolves) all branch and CR logical instructions.
- Two simple units (SU1 and SU2)
  - Add and subtract
  - Shift and rotate
  - Logical operations
  - Support for 64-bit SPE instructions in SU1
- Multiple-cycle unit (MU)—The MU is shown in [Figure 5-3](#).

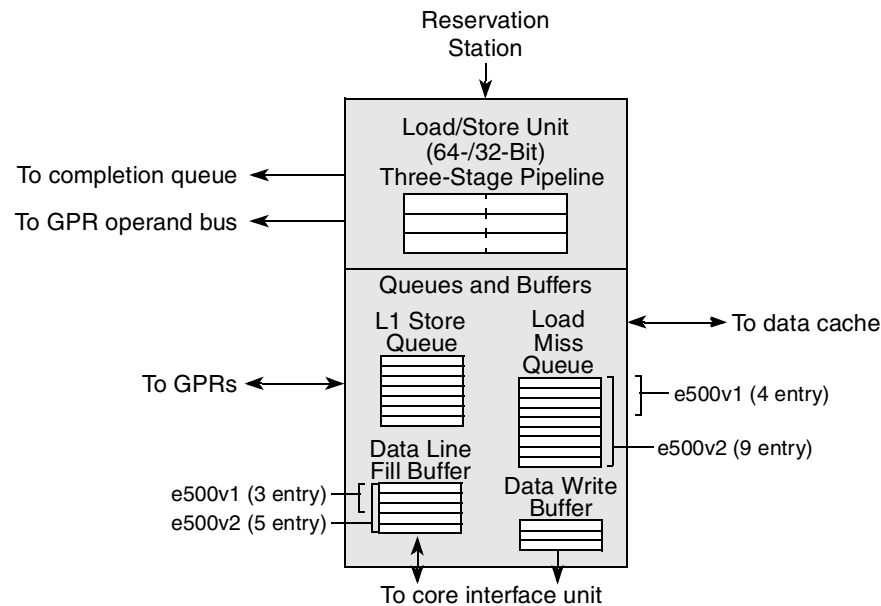


**Figure 5-3. Four-Stage MU Pipeline, Showing Divide Bypass**

The MU has the following features:

- Four-cycle latency for all multiplication, including SPE integer and fractional multiply instructions and embedded scalar and vector floating-point multiply instructions
- Variable-latency divide: 4, 11, 19, and 35 cycles for all integer divide instructions. If **rA** or **rB** is zero, floating-point divide instructions take 4 cycles; all others take 29. Note that although most divide instructions take more than 4 cycles to execute, the MU allows subsequent multiply instructions to execute through all four MU stages in parallel with the divide.

- 4-cycle floating-point add and subtract
- The load/store unit (LSU) is shown in [Figure 5-4](#).



**Figure 5-4. Three-Stage Load/Store Unit**

The LSU has the following features:

- Three-cycle load latency
- Fully pipelined
- Load miss queue allows up to four load misses before stalling (up to nine load misses in the e500v2).
- Load hits can continue to be serviced when the load miss queue is full.
- The seven-entry L1 store queue allows full pipelining of stores.
- The three-entry data line fill buffer (five-entry on the e500v2) is used for loads and cacheable stores. Stores are allocated here so loads can access data from the store immediately.
- The data write buffer contains three entries: one dedicated for snoop pushes, one dedicated for castouts, and one that can be used for snoop pushes or cast outs.
- Cache coherency
  - Supports four-state cache coherency: modified-exclusive, exclusive, shared, and invalid (MESI). Note, however that shared state may not be accessible in some implementations.
  - Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)—internal bus
  - High-speed, on-chip local bus with data tagging
  - 32-bit address bus
  - Address protocol with address pipelining and retry/copyback derived from bus used by previous generations of processors (referred to as the 60x bus)

- Two general-purpose read data buses and one write data bus
- Extended exception handling
  - Supports embedded category interrupt model
    - Less than 10-cycle interrupt latency
    - Interrupt vector prefix register (IVPR)
    - Interrupt vector offset registers (IVORs) 0–15 and 32–35
    - Exception syndrome register (ESR)
    - Preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfei** instruction
    - A separate set of resources for machine-check interrupts
    - SPE unavailable exception
    - Floating-point data exception
    - Floating-point round exception
    - Performance monitor
- Memory management unit (MMU)
  - 32-bit effective address translated to 32-bit real address (using a 41-bit interim virtual address) for the e500v1 core and 36-bit real addressing for the e500v2 core
  - TLB entries for variable- (4-Kbyte–256-Mbyte pages for the e500v1 and 4-Kbyte–4-Gbyte pages for the e500v2) and fixed-size (4-Kbyte) pages
  - Data L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Instruction L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Unified L2 MMU
    - 16-entry, fully-associative TLB array for variable-sized pages
    - e500v1—A 256-entry, 2-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
    - e500v2—A 512-entry, 4-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
  - Software reload for TLBs
  - Virtual memory support for as much as 4 Gbytes ( $2^{32}$ ) of effective address space
  - Real memory support for as much as 4 Gbytes ( $2^{32}$ ) of physical memory on the e500v1 and 64 Gbytes ( $2^{36}$ ) on the e500v2
  - Support for big-endian and true little-endian memory on a per-page basis
- Power management
  - Low-power design

- Power-saving modes: core-halted and core-stopped
- Internal clock multipliers ranging from 1 to 8 times the bus clock, including integer and half-mode multipliers. The MPC8533E supports multipliers of 2, 2.5, 3, and 3.5
- Dynamic power management of execution units, caches, and MMUs
- NAP, DOZE, and SLEEP bits in HID0 can be used to assert *nap*, *doze*, and *sleep* output signals to initiate power-saving modes at the integrated device level.
- Testability
  - LSSD scan design
  - JTAG interface
  - ESP support
- Reliability and serviceability
  - Parity checking on caches
  - Parity checking on e500 local bus

### 5.3.1 e500v2 Differences

The e500v2 provides the following additional features not supported by the e500v1:

- The e500v2 uses 36-bit physical addressing, which is supported by the following:
  - MMU assist register 7 (MAS7)
  - HID0[EN\_MAS7\_UPDATE]
  - Programmable jumper options to specify the upper bits of the reset vector.
- The e500v2 has a 512-entry, 4-way set-associative unified TLB for TLB1.
- The maximum variable page size is extended to 4 Gbytes.
- Embedded double-precision floating-point support has been added. These instructions use the 64-bit GPRs as single, 64-bit double-precision operands. This functionality is enabled through MSR[SPE].
- Slightly different functionality of HID1[RFXE] bit.
- The data line fill buffer in the LSU is expanded from three to five entries.
- The load miss queue in the LSU is expanded from four to nine entries.
- TBSEL and TBEE bits have been added to the performance monitor global control register 0 (PMGC0) to support monitoring of time base events.
- Minor modifications to the SPE instruction set.
- Data cache flush assist capability, supported through HID0[DCFA]. When DCFA is set, the cache miss replacement algorithm ignores invalid entries and follows the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed load or **dcbz** instructions required to flush the cache.

Detailed descriptions of these differences are provided in their respective chapters.

**NOTE**

Unless otherwise indicated, references to e500 apply to both e500v1 and e500v2.

**5.4 Instruction Set**

The e500 implements the following instructions:

- The embedded category instruction set for 32-bit implementations. This is composed primarily of the user-level instructions defined by the Power Architecture user instruction set architecture (UISA). The e500 does not include floating-point instructions that require floating-point registers (FPRs), load string, or store string instructions.
- The e500 supports the following instructions:
  - Integer select. Now part of the base category. Consists of the Integer Select instruction (**isel**), which functions as an if-then-else statement that selects between two source registers by comparison to a CR bit. This instruction eliminates conditional branches, decreases latency, and reduces the code footprint.
  - Performance monitor. [Table 5-2](#) lists performance monitor instructions.

**Table 5-2. Performance Monitor Instructions**

Name	Mnemonic	Syntax
Move from Performance Monitor Register	<b>mfpmr</b>	rD,PMRN
Move to Performance Monitor Register	<b>mtpmr</b>	PMRN,rS

- Cache locking. Consists of the instructions described in [Table 5-3](#).

**Table 5-3. Cache Locking Instructions**

Name	Mnemonic	Syntax
Data Cache Block Lock Clear	<b>dcblc</b>	CT, rA, rB
Data Cache Block Touch and Lock Set	<b>dcbtls</b>	CT, rA, rB
Data Cache Block Touch for Store and Lock Set	<b>dcbtstls</b>	CT, rA, rB
Instruction Cache Block Lock Clear	<b>icblc</b>	CT, rA, rB
Instruction Cache Block Touch and Lock Set	<b>icbtls</b>	CT, rA, rB

- Machine check. Defines the Return from Machine Check Interrupt instruction (**rfmci**).
- SPE vector instructions. Vector instructions are defined that view the 64-bit GPRs as composed of a vector of two 32-bit elements (some instructions also read or write 16-bit elements). Some scalar instructions produce a 64-bit scalar result.
- The embedded floating-point categories provide scalar and vector floating-point instructions. Scalar single-precision floating-point instructions use only the lower 32 bits of the GPRs; double-precision operands (e500v2 only) use all 64 bits. [Table 5-4](#) lists embedded floating-point instructions.

**Table 5-4. Scalar and Vector Embedded Floating-Point Instructions**

Instruction	Mnemonic			Syntax
	Scalar SP	Scalar DP	Vector	
Convert Floating-Point Single- from Double-Precision	—	efscfd	—	rD,rB
Convert Floating-Point Double- from Single-Precision	—	efdcfs	—	rD,rB
Convert Floating-Point from Signed Fraction	efscfsf	efdcfsf	evfscfsf	rD,rB
Convert Floating-Point from Signed Fraction	efscfsf	efdcfsf	evfscfsf	rD,rB
Convert Floating-Point from Signed Integer	efscfsi	efdcfsi	evfscfsi	rD,rB
Convert Floating-Point from Unsigned Fraction	efscfuf	efdcfuf	evfscfuf	rD,rB
Convert Floating-Point from Unsigned Integer	efscfui	efdcfui	evfscfui	rD,rB
Convert Floating-Point to Signed Fraction	efscfsf	efdcfsf	evfscfsf	rD,rB
Convert Floating-Point to Signed Integer	efscfsi	efdcfsi	evfscfsi	rD,rB
Convert Floating-Point to Signed Integer with Round toward Zero	efscfsiz	efdcfsiz	evfscfsiz	rD,rB
Convert Floating-Point to Unsigned Fraction	efscfuf	efdcfuf	evfscfuf	rD,rB
Convert Floating-Point to Unsigned Integer	efscfui	efdcfui	evfscfui	rD,rB
Convert Floating-Point to Unsigned Integer with Round toward Zero	efscfuiiz	efdcfuiiz	evfscfuiiz	rD,rB
Floating-Point Absolute Value	efsabs	efdabs	evfsabs	rD,rA
Floating-Point Add	efsadd	efdadd	evfsadd	rD,rA,rB
Floating-Point Compare Equal	efscmpeq	efdcmpeq	evfscmpeq	crD,rA,rB
Floating-Point Compare Greater Than	efscmpgt	efdcmpgt	evfscmpgt	crD,rA,rB
Floating-Point Compare Less Than	efscmplt	efdcmplt	evfscmplt	crD,rA,rB
Floating-Point Divide	efdiv	efddiv	evfdiv	rD,rA,rB
Floating-Point Multiply	efsmul	efdmul	evfsmul	rD,rA,rB
Floating-Point Negate	efsneg	efdneg	evfsneg	rD,rA
Floating-Point Negative Absolute Value	efsnabs	efdnabs	evfsnabs	rD,rA
Floating-Point Subtract	efssub	efdsb	evfssub	rD,rA,rB
Floating-Point Test Equal	efststeq	efdtsteq	evfststeq	crD,rA,rB
Floating-Point Test Greater Than	efststgt	efdtstgt	evfststgt	crD,rA,rB
Floating-Point Test Less Than	efststlt	efdtstlt	evfststlt	crD,rA,rB

- BTB locking instructions. The core complex provides a 512-entry BTB for efficient processing of branch instructions. The BTB is a branch target address cache, organized as 128 rows with 4-way set associativity, that holds the address and target instruction of the 512 most-recently taken branches. [Table 5-5](#) lists BTB instructions.



Table 5-5. BTB Locking Instructions

Name	Mnemonic	Syntax
Branch Buffer Load Entry and Lock Set	<b>bblels</b>	—
Branch Buffer Entry Lock Reset	<b>bbelr</b>	—

## 5.5 Instruction Flow

The e500 core is a pipelined, superscalar processor with parallel execution units that allow instructions to execute out of order but record their results in order. Pipelining breaks instruction processing into discrete stages, so multiple instructions in an instruction sequence can occupy the successive stages: as an instruction completes one stage, it passes to the next, leaving the previous stage available to a subsequent instruction. So, even though it may take multiple cycles for an instruction to pass through all of the pipeline stages, once a pipeline is full, instruction throughput is much shorter than the latency.

A superscalar processor is one that issues multiple independent instructions into separate execution units, allowing parallel execution. The e500 core has five execution units, one each for branch (BU), load/store (LSU), and multiple-cycle operations (MU), and two for simple arithmetic operations (SU1 and SU2). The MU and SU1 arithmetic execution units also execute 64-bit SPE vector instructions, using both the lower and upper halves of the 64-bit GPRs.

The parallel execution units allow multiple instructions to execute in parallel and out of order. For example, a low-latency addition instruction that is issued to an SU after an integer divide is issued to the MU should finish executing before the higher latency divide instruction. The add instruction can make its results available to a subsequent instruction, but it cannot update the architected GPR specified as its target operand ahead of the multiple-cycle divide instruction.

### 5.5.1 Initial Instruction Fetch

The e500 core begins execution at fixed virtual address 0xFFFF\_FFFC. The MMU has a default page translation which maps this to the identical physical address. So, the instruction at physical address 0xFFFF\_FFFC must be a branch to another address within the 4-Kbyte boot page.

### 5.5.2 Branch Detection and Prediction

To improve branch performance, the e500 provides implementation-specific dynamic branch prediction using the BTB to resolve branch instructions and improve the accuracy of branch predictions. Each of the 512 entries in the 4-way set associative address cache of branch target addresses includes a 2-bit saturating branch history counter, whose value is incremented or decremented depending on whether the branch was taken. These bits can take on four values indicating strongly taken, weakly taken, weakly not taken, and strongly not taken. The BTB is used not only to predict branches, but to detect branches during the fetch stage, offering an efficient way to access instruction streams for branches predicted as taken.

In the e500, all branch instructions are assigned positions in the completion queue at dispatch. Speculative instructions in branch target streams are allowed to execute and proceed through the completion queue, although they can complete only after the branch prediction is resolved as correct and after the branch instruction itself completes.

If a branch resolves as correct, instructions in the target stream are marked nonspeculative and are allowed to complete. If the branch history bits in the BTB indicated weakly taken or weakly not taken, the prediction is upgraded to strongly taken or strongly not taken.

If a branch resolves as incorrect, instructions in the target stream are flushed from the execution pipeline, the branch history bits are updated in the BTB entry, and nonspeculative fetching begins from the correct path.

### 5.5.3 e500 Execution Pipeline

The seven stages of the e500 execution pipeline—fetch1, fetch2/predecode, decode/dispatch, issue, execute, complete, and write back—are highlighted in grey in Figure 5-5.

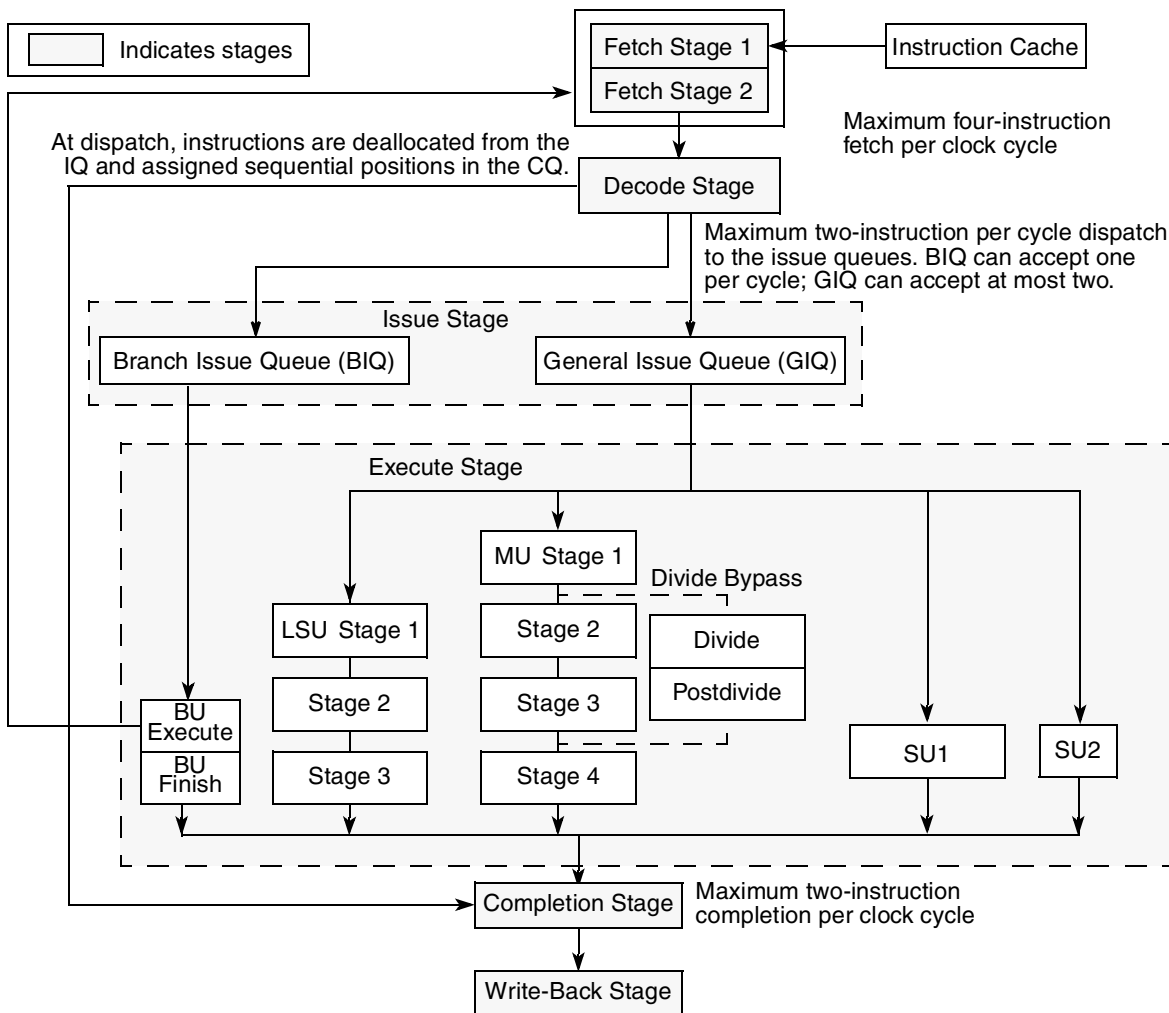


Figure 5-5. Instruction Pipeline Flow

The common pipeline stages are as follows:

- Instruction fetch—Includes the clock cycles necessary to request an instruction and the time the memory system takes to respond to the request. Instructions retrieved are latched into the instruction queue (IQ) for subsequent consideration by the dispatcher.

Instruction fetch timing depends on many variables, such as whether an instruction is in the on-chip instruction cache or an L2 cache (if implemented). Those factors increase when it is necessary to fetch instructions from system memory and include the processor-to-bus clock ratio, the amount of bus traffic, and whether any cache coherency operations are required.

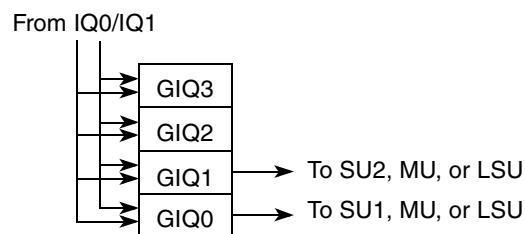
Because there are so many variables, unless otherwise specified, the instruction timing examples in this chapter assume optimal performance and show the portion of the fetch stage in which the instruction is in the instruction queue. The fetch1 and fetch2 stages are primarily involved in retrieving instructions.

- The decode/dispatch stage fully decodes each instruction; most instructions are dispatched to the issue queues (however, **isync**, **rfi**, **sc**, **nops**, and some other instructions do not go to issue queues).
- The two issue queues, BIQ and GIQ, can accept as many as one and two instructions, respectively, in a cycle. The behavior of instruction dispatch is covered in significant detail in the *e500 Software Optimization Guide*. The following simplification covers most cases:
  - Instructions dispatch only from the two lowest IQ entries—IQ0 and IQ1.
  - A total of two instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the CQ for an instruction to decode and dispatch (this includes instructions that are assigned a space in the CQ but not in an issue queue).

Dispatch is treated as an event at the end of the decode stage. The issue stage reads source operands from rename registers and register files and determines when instructions are latched into the execution unit reservation stations. Note that the e500 has 14 rename registers, one for each completion queue entry, so instructions cannot stall because of a shortage of rename registers.

The general behavior of the two issue queues is described as follows:

- The GIQ accepts as many as two instructions from the dispatch unit per cycle. SU1, SU2, MU, and all LSU instructions (including 64-bit loads and stores) are dispatched to the GIQ, shown in [Figure 5-6](#).



**Figure 5-6. GPR Issue Queue (GIQ)**

Instructions can be issued out-of-order from the bottom two GIQ entries (GIQ1–GIQ0). GIQ0 can issue to SU1, MU, and LSU. GIQ1 can issue to SU2, MU, and LSU.

Note that SU2 executes a subset of the instructions that can be executed in SU1. The ability to identify and dispatch instructions to SU2 increases the availability of SU1 to execute more computational-intensive instructions.

An instruction in GIQ1 destined for SU2 or the LSU need not wait for an MU instruction in GIQ0 that is stalled behind a long-latency divide.

- The execute stage accepts instructions from its issue queue when the appropriate reservation stations are not busy. In this stage, the operands assigned to the execution stage from the issue stage are latched.

The execution unit executes the instruction (perhaps over multiple cycles), writes results on its result bus, and notifies the CQ when the instruction finishes. The execution unit reports any exceptions to the completion stage. Instruction-generated exceptions are not taken until the excepting instruction is next to retire.

Most integer instructions have a 1-cycle latency, so results of these instructions are available 1 clock cycle after an instruction enters the execution unit. The MU and LSU are pipelined, as shown in [Figure 5-5](#).

Branches resolve in execute stage. If a branch is mispredicted, it takes 5 cycles for the next instruction to reach the execute stage.

- The complete and write-back stages maintain the correct architectural machine state and commit results to the architecture-defined registers in the proper order. If completion logic detects an instruction containing an exception status or a mispredicted branch, all following instructions are cancelled, their execution results in rename registers are discarded, and the correct instruction stream is fetched.

The complete stage ends when the instruction is retired. Two instructions can be retired per clock cycle. If no dependencies exist, as many as two instructions are retired in program order.

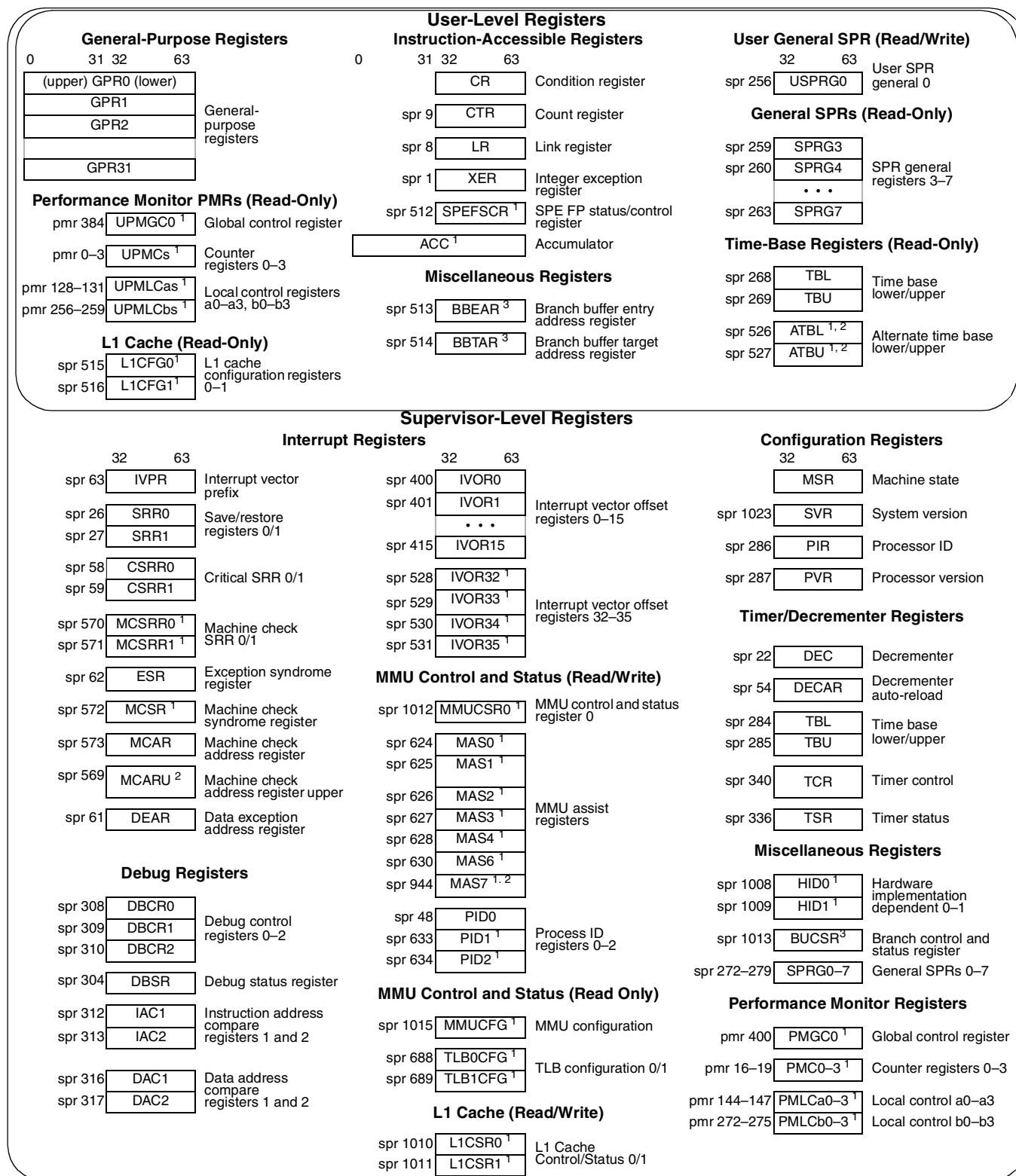
The write-back stage occurs in the clock cycle after the instruction is retired.

The e500 core also provides new instructions that perform single-instruction, multiple-data (SIMD) operations. These signal processing instructions consist of parallel operations on both the upper and lower 32 bits of two 64-bit GPR values and produce two 32-bit results written to a 64-bit GPR.

As shown in [Figure 5-5](#), the LSU, MU, and SU1 replicate logic to support 64-bit operations. Although a vector instruction generates separate, discrete results in the upper and lower halves of the target GPR, latency and throughput for vector instructions are the same as those for their scalar equivalents.

## 5.6 Programming Model

The following section describes the e500 core registers. [Figure 5-7](#) shows the e500 register set.



<sup>1</sup> These registers are defined by the EIS  
<sup>2</sup> e500v2 only  
<sup>3</sup> These registers are e500-specific

Figure 5-7. e500 Core Programming Model

## 5.7 On-Chip Cache Implementation

The core complex contains separate 32-Kbyte, eight-way set-associative, level 1 (L1) instruction and data caches to give rapid access to instructions and data.

The data cache supports four-state MESI memory coherency protocol. The core complex broadcasts all cache management functions based on the setting of the address broadcast enable bit, `HID1[ABE]`, allowing management of other caches in the system.

MPC8533E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

The caches implement a pseudo-least-recently-used (PLRU) replacement algorithm.

Parity generation and checking may be enabled for both caches, and each cache can be independently invalidated through `L1CSR1` and `L1CSR0`. Additionally, instructions are provided to perform cache locking and unlocking on both data and instruction caches on a cache-block granularity. These are listed in [Section 5.10.3, “Cache Control Instructions.”](#)

Individual instruction cache blocks and data cache blocks can be invalidated using the `icbi` and `dcbi` instructions, respectively. The entire data cache can be invalidated by setting `L1CSR0[CFI]`; the entire instruction cache can be invalidated by setting `L1CSR1[ICFI]`.

## 5.8 Interrupts and Exception Handling

The e500 core supports an extended exception handling model, with nested interrupt capability and extensive interrupt vector programmability. The following sections define the exception model, including an overview of exception handling as implemented on the e500 core, a brief description of the exception classes, and an overview of the registers involved in the processes.

### 5.8.1 Exception Handling

In general, interrupt processing begins with an exception that occurs due to external conditions, errors, or program execution problems. When the exception occurs, the processor checks to verify interrupt processing is enabled for that particular exception. If enabled, the interrupt causes the state of the processor to be saved in the appropriate registers and prepares to begin execution of the handler located at the associated vector address for that particular exception.

Once the handler is executing, the implementation may need to check one or more bits in the exception syndrome register (ESR) or the `SPEFSCR`, depending on the exception, to verify the specific cause of the exception and take appropriate action.

The core complex provides the interrupts described in [Section 5.8.5, “Interrupt Registers.”](#)

## 5.8.2 Interrupt Classes

All interrupts may be categorized as asynchronous/synchronous and critical/noncritical.

- Asynchronous interrupts (such as machine check, critical input, and external interrupts) are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported in a save/restore register is the address of the instruction that would have executed next had the asynchronous interrupt not occurred.
- Synchronous interrupts are those that are caused directly by the execution or attempted execution of instructions. Synchronous inputs may be either precise or imprecise, which are described as follows:
  - Synchronous precise interrupts are those that precisely indicate the address of the instruction causing the exception that generated the interrupt or, in some cases, the address of the immediately following instruction. The interrupt type and status bits indicate which instruction is addressed in the appropriate save/restore register.
  - Synchronous imprecise interrupts are those that may indicate the address of the instruction causing the exception that generated the interrupt or some instruction after the instruction causing the interrupt. If the interrupt was caused by either the context synchronizing mechanism or the execution synchronizing mechanism, the address in the appropriate save/restore register is the address of the interrupt forcing instruction. If the interrupt was not caused by either of those mechanisms, the address in the save/restore register is the last instruction to start execution and may not have completed. No instruction following the instruction in the save/restore register has executed.

## 5.8.3 Interrupt Types

The e500 core processes all interrupts as either machine check, critical, or noncritical types. Separate control and status register sets are provided for each interrupt type. The core handles interrupts from these three types in the following priority order:

1. Machine check interrupt (highest priority)—The e500 defines a separate set of resources for the machine check interrupt. They use the machine check save and restore registers (MCSRR0/MCSRR1) to save state when they are taken, and they use the **rfmci** instruction to restore state. These interrupts can be masked by the machine check enable bit, MSR[ME].
2. Noncritical interrupts—First-level interrupts that allow the processor to change program flow to handle conditions generated by external signals, errors, or unusual conditions arising from program execution or from programmable timer-related events. These interrupts are largely identical to those previously defined by the OEA portion of the architecture. They use save and restore registers (SRR0/SRR1) to save state when they are taken and they use the **rfi** instruction to restore state. Asynchronous noncritical interrupts can be masked by the external interrupt enable bit, MSR[EE].
3. Critical interrupts—Critical interrupts can be taken during a noncritical interrupt or during regular program flow. They use the critical save and restore registers (CSRR0/CSRR1) to save state when they are taken and they use the **rfci** instruction to restore state. These interrupts can be masked by

the critical enable bit, MSR[CE]. The embedded category defines the critical input, watchdog timer, and machine check interrupts as critical interrupts, but the e500 implements a third set of resources for the machine check interrupt, as described in [Table 5-6](#).

All interrupts except machine check are ordered within the two categories of noncritical and critical, such that only one interrupt of each category is reported, and when it is processed (taken), no program state is lost. Because save/restore register pairs are serially reusable, program state may be lost when an unordered interrupt is taken.

## 5.8.4 Upper Bound on Interrupt Latencies

Core complex interrupt latency is defined as the number of core clocks between the sampling of the interrupt signal as asserted and the initiation of the IVOR fetch (that is, the fetch of the first instruction in the handler). Core complex interrupt latency is determinate unless a guarded load or a cache-inhibited **stwxc** is being executed, in which case the latency is indeterminate. The minimum latency is 3 core clocks and the maximum is 8, not including the 2 bus clock cycles required to synchronize the interrupt signal from the pad.

When an interrupt is taken, all instructions in the IQ are thrown away unless the oldest instruction is a load/store instruction. That is, if an asynchronous interrupt is being serviced and the oldest instruction is not a load/store instruction, the core complex goes straight from sampling the interrupt to ensuring a recoverable state and issuing an exception. If a load/store instruction is oldest, the core complex waits 4 clocks before ensuring a recoverable state. During this time, any instruction finished by the LSU is deallocated.

## 5.8.5 Interrupt Registers

The registers associated with interrupt and exception handling are described in [Table 5-6](#).

**Table 5-6. Interrupt Registers**

Register	Description
<b>Noncritical Interrupt Registers</b>	
SRR0	Save/restore register 0—Holds the address of the instruction causing the exception or the address of the instruction that will execute after the <b>rfi</b> instruction.
SRR1	Save/restore register 1—Holds machine state on noncritical interrupts and restores machine state after an <b>rfi</b> instruction is executed.
<b>Critical Interrupt Registers</b>	
CSRR0	Critical save/restore register 0—On critical interrupts, holds either the address of the instruction causing the exception or the address of the instruction that will execute after the <b>rfci</b> instruction.
CSRR1	Critical save/restore register 1—Holds machine state on critical interrupts and restores machine state after an <b>rfci</b> instruction is executed.
<b>Machine Check Interrupt Registers</b>	
MCSRR0	Machine check save/restore register 0—Used to store the address of the instruction that will execute after an <b>rfmci</b> instruction is executed.
MCSRR1	Machine check save/restore register 1—Holds machine state on machine check interrupts and restores machine state (if recoverable) after an <b>rfmci</b> instruction is executed.



Table 5-6. Interrupt Registers (continued)

Register	Description
MCAR	Machine check address register—Holds the address of the data or instruction that caused the machine check interrupt. MCAR contents are not meaningful if a signal triggered the machine check interrupt.
<b>Syndrome Registers</b>	
MCSR	Machine check syndrome register—Holds machine state information on machine check interrupts and restores machine state after an <b>rfmci</b> instruction is executed.
ESR	Exception syndrome register—Provides a syndrome to differentiate between the different kinds of exceptions that generate the same interrupt type. Upon generation of a specific exception type, the associated bit is set and all other bits are cleared.
<b>SPE Interrupt Registers</b>	
SPEFSCR	Signal processing and embedded floating-point status and control register—Provides interrupt control and status as well as various condition bits associated with the operations performed by the SPE.
<b>Other Interrupt Registers</b>	
DEAR	Data exception address register—Holds the address that was referenced by a load, store, or cache management instruction that caused an alignment, data TLB miss, or data storage interrupt.
IVPR IVORs	Together, IVPR[32–47]    IVOR $n$ [48–59]    0b0000 define the address of an interrupt-processing routine. See <a href="#">Table 5-7</a> and the EREF for more information.

Each interrupt has an associated interrupt vector address, obtained by concatenating the IVPR value with the address index in the associated IVOR (that is, IVPR[32–47] || IVOR $n$ [48–59] || 0b0000). The resulting address is that of the instruction to be executed when that interrupt occurs. IVPR and IVOR values are indeterminate on reset, and must be initialized by the system software using **mtspr**. [Table 5-7](#) lists IVOR registers implemented on the e500 and the associated interrupts.

Table 5-7. Interrupt Vector Registers and Exception Conditions

Register	Interrupt
<b>Embedded Category–Defined IVORs</b>	
IVOR0	Critical input
IVOR1	Machine check interrupt offset
IVOR2	Data storage interrupt offset
IVOR3	Instruction storage interrupt offset
IVOR4	External input interrupt offset
IVOR5	Alignment interrupt offset
IVOR6	Program interrupt offset
IVOR7	Floating-point unavailable interrupt offset
IVOR8	System call interrupt offset
IVOR9	Auxiliary processor unavailable interrupt offset
IVOR10	Decrementer interrupt offset
IVOR11	Fixed-interval timer interrupt offset
IVOR12	Watchdog timer interrupt offset
IVOR13	Data TLB error interrupt offset

**Table 5-7. Interrupt Vector Registers and Exception Conditions (continued)**

Register	Interrupt
IVOR14	Instruction TLB error interrupt offset
IVOR15	Debug interrupt offset
<b>e500-Specific IVORs</b>	
IVOR32	SPE unavailable interrupt offset
IVOR33	SPE floating-point data exception interrupt offset
IVOR34	SPE floating-point round exception interrupt offset
IVOR35	Performance monitor

## 5.9 Memory Management

The e500 core complex supports demand-paged virtual memory as well other memory management schemes that depend on precise control of effective-to-physical address translation and flexible memory protection as defined by the architecture. The mapping mechanism consists of software-managed TLBs that support variable-sized pages with per-page properties and permissions. The following properties can be configured for each TLB:

- User-mode page execute access
- User-mode page read access
- User-mode page write access
- Supervisor-mode page execute access
- Supervisor-mode page read access
- Supervisor-mode page write access
- Write-through required (W)
- Caching inhibited (I)
- Memory coherency required (M) (the M bit has no effect on the MPC8533E). See [Table 5-8](#) (feature “Multiprocessor functionality”) for further details.
- Guarded (G)
- Endianness (E)
- User-definable (U0–U3), a 4-bit implementation-specific field

The core complex employs a two-level memory management unit (MMU) architecture. There are separate instruction and data level-1 (L1) MMUs backed up by a unified level-2 (L2) MMU.

This two-level structure is shown in Figure 5-8.

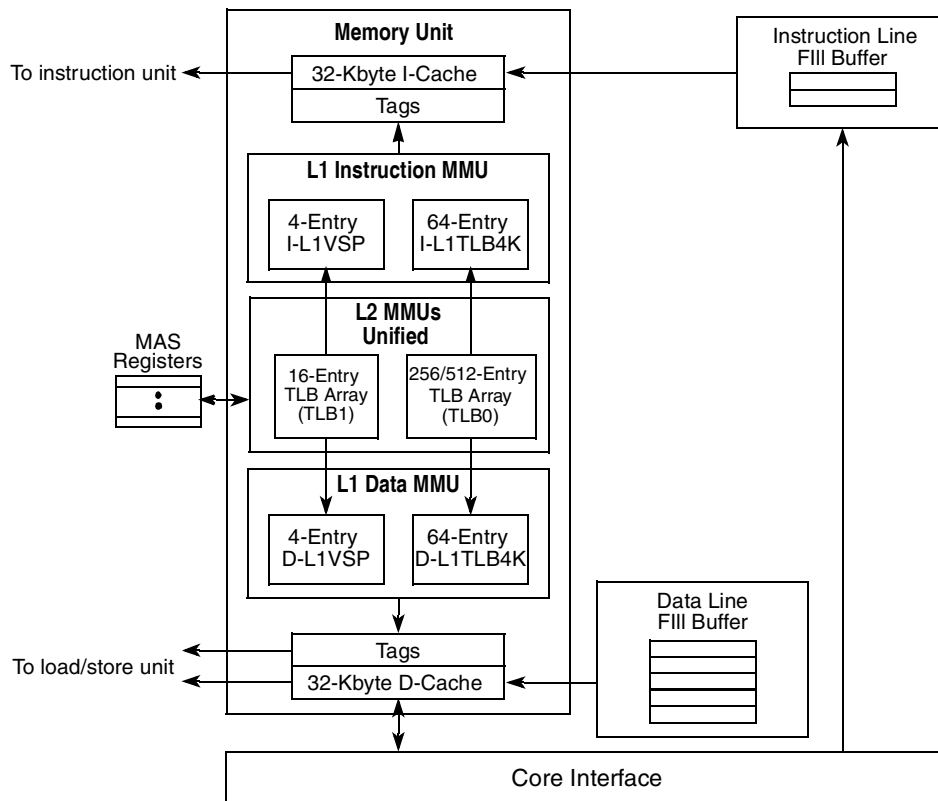


Figure 5-8. MMU Structure

Level-1 MMUs have the following features:

- Four-entry, fully associative TLB array that supports all nine page sizes
- 64-entry, 4-way set-associative TLB 4-Kbyte array that supports 4-Kbyte pages only
- Hardware partially managed by L2 MMU
- Supports snooping of TLBs by both internal and external **tlbivax** instructions

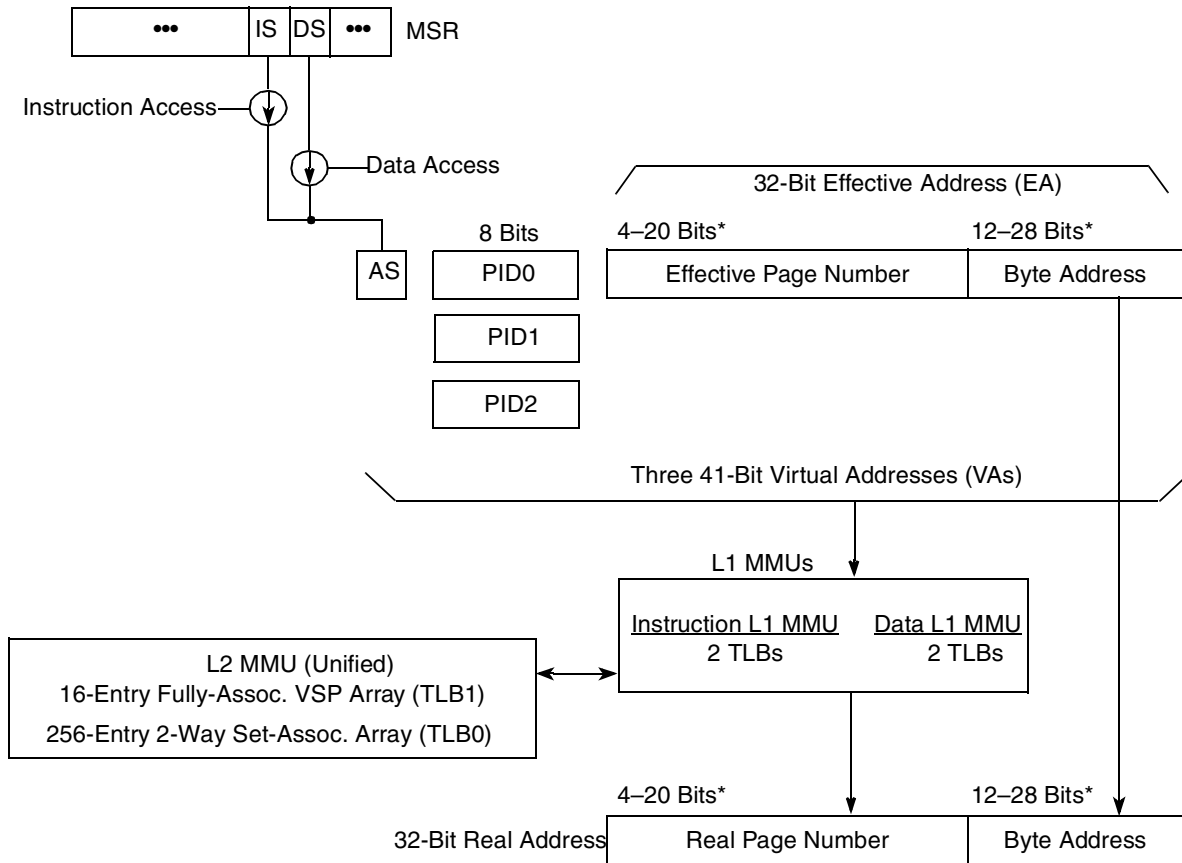
The level-2 MMU has the following features:

- A 16-entry, fully associative L2 TLB array (TLB1) that supports all nine variable page sizes
- TLB array (TLB0) that supports only 4-Kbyte pages, as follows:
  - e500v1—256-entry, 2-way set-associative TLB array
  - e500v2—512-entry, 4-way set-associative TLB array
- Hardware assist for TLB miss exceptions
- Software managed by **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, **tlbivax**, and **mtspr** instructions
- Supports snooping of TLB by both internal and external **tlbivax** instructions

### 5.9.1 Address Translation

The core complex fetch and load/store units generate 32-bit effective addresses. The MMU translates these addresses to real addresses (32-bit real addresses for the e500v1 core, 36-bit for the e500v2) (which are used for memory bus accesses) using an interim 41-bit virtual address.

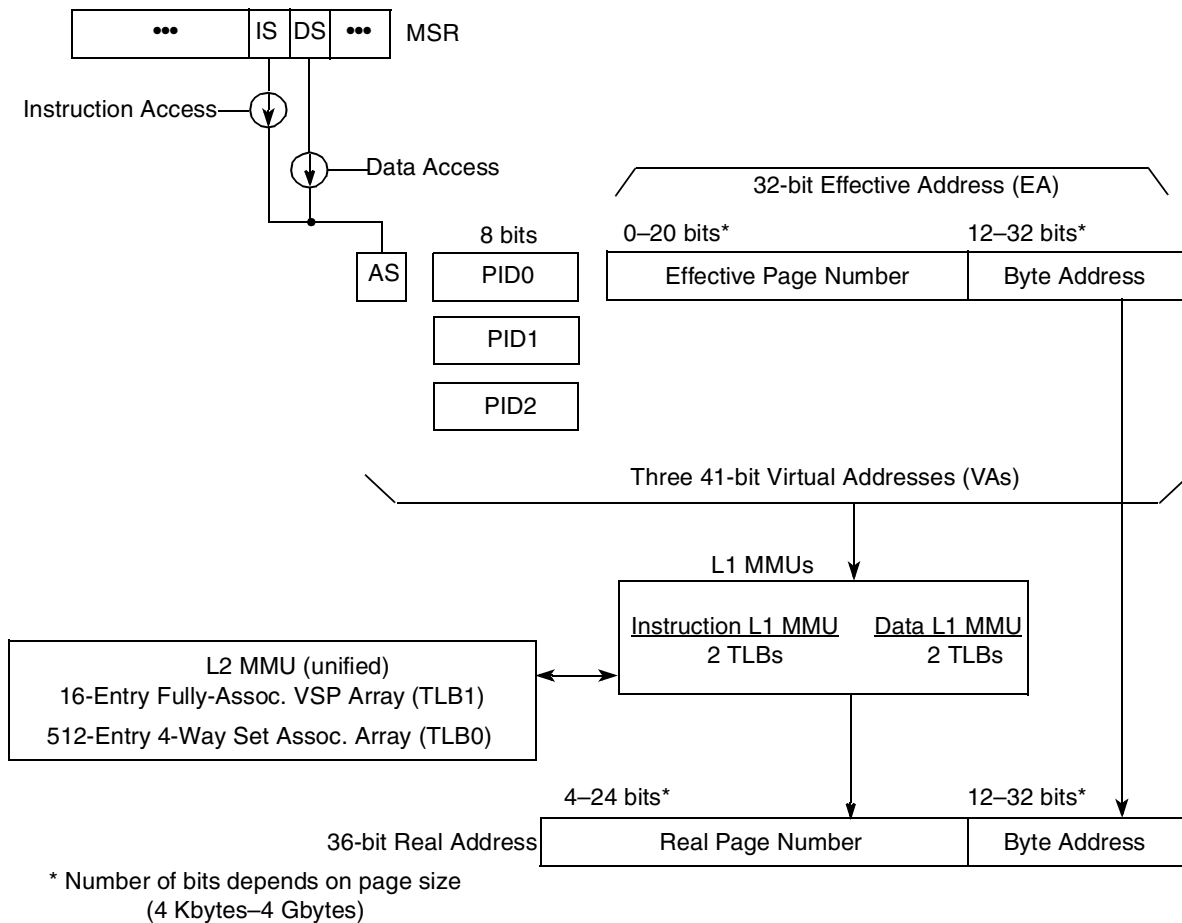
Figure 5-9 shows the translation flow for the e500v1 core.



\* Number of bits depends on page size (4 Kbytes–256 Mbytes).

**Figure 5-9. Effective-to-Real Address Translation Flow**

Figure 5-10 shows the same translation flow for the e500v2 core.



**Figure 5-10. Effective-to-Real Address Translation Flow (e500v2)**

The appropriate L1 MMU (instruction or data) is checked for a matching address translation. The instruction L1 MMU and data L1 MMU operate independently and can be accessed in parallel, so that hits for instruction accesses and data accesses can occur in the same clock. If an L1 MMU misses, the request for translation is forwarded to the unified (instruction and data) L2 MMU. If found, the contents of the TLB entry are concatenated with the byte address to obtain the physical address of the requested access. On misses, the L1 TLB entries are replaced from their L2 TLB counterparts using a true LRU algorithm.

## 5.9.2 MMU Assist Registers (MAS0–MAS4 and MAS6–MAS7)

MMU assist registers are used to hold values either read from or to be written to the TLBs and information required to identify the TLB to be accessed. MAS3 implements the real page number (RPN), the user attribute bits (U0–U3), and permission bits (UX, SX, UW, SW, UR, SR) that specify user and supervisor read, write, and execute permissions.

The e500 does not implement MAS5.

MAS registers are affected by the following instructions (see the EREF for more detailed information):

- MAS registers are accessed with the **mtspr** and **mfspir** instructions.
- The TLB Read Entry instruction (**tlbre**) causes the contents of a single TLB entry from the L2 MMU to be placed in defined locations in MAS0–MAS3 (and optionally MAS7 on the e500v2). The TLB entry to be extracted is determined by information written to MAS0 and MAS2 before the **tlbre** instruction is executed.
- The TLB Write Entry instruction (**tlbwe**) causes the information stored in certain locations of MAS0–MAS3 (and MAS7 on the e500v2) to be written to the TLB specified in MAS0.
- The TLB Search Indexed instruction (**tlbsx**) updates MAS registers conditionally, based on success or failure of a lookup in the L2 MMU. The lookup is specified by the instruction encoding and specific search fields in MAS6. The values placed in the MAS registers may differ, depending on a successful or unsuccessful search.

For TLB miss and certain MMU-related DSI/ISI exceptions, MAS4 provides default values for updating MAS0–MAS2.

### 5.9.3 Process ID Registers (PID0–PID2)

The e500 core complex also implements three process ID (PID) registers that hold the values used to construct the three virtual addresses for each access. These process IDs provide an extended page sharing capability. Which of these three virtual addresses is used is controlled by the TID field of a matching TLB entry, and when TID = 0x00 (identifying a page as globally shared), the PID values are ignored.

A hit to multiple TLB entries in the L1 MMU (even if they are in separate arrays) or a hit to multiple entries in the L2 MMU is considered to be a programming error.

### 5.9.4 TLB Coherency

The core complex provides the ability to invalidate a TLB entry, as defined by the architecture. The **tlbivax** instruction invalidates a matching local TLB entry. Execution of this instruction is also broadcast on the core complex bus (CCB) if HID1[ABE] is set. The core complex also snoops TLB invalidate transactions on the CCB from other bus masters.

On the MPC8533E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

## 5.10 Memory Coherency

The core complex supports four-state memory coherency. Memory coherency is hardware-supported on the system bus through bus snooping and the retry/copyback bus protocol, and through broadcasting of cache management instructions. Translation coherency is also hardware-supported through broadcasting and bus snooping of TLB invalidate transactions. The four-state MESI protocol supports efficient large-scale real-time data sharing between multiple caching bus masters.

### 5.10.1 Atomic Update Memory References

The e500 core supports atomic update memory references for both aligned word forms of data using the load and reserve and store conditional instruction pair, **lwarx** and **stwcx**. Typically, a load and reserve instruction establishes a reservation and is paired with a store conditional instruction to achieve the atomic operation. However, there are restrictions and requirements for this functionality. The processor revokes reservations during a context switch, so the programmer must reacquire the reservation after a context switch occurs.

### 5.10.2 Memory Access Ordering

The core complex supports weakly ordered references to memory. Thus the e500 manages the order and synchronization of instructions to ensure proper execution when memory is shared between multiple processes or programs. The cache and data memory control attributes, along with **msync** and **mbar**, provide the required access control.

### 5.10.3 Cache Control Instructions

The core complex supports instructions for performing a full range of cache control functions, including cache locking by line. The core complex supports broadcasting and snooping of these cache control instructions on the CCB. The e500 core also supports the following e500-specific cache locking instructions:

- Data Cache Block Lock Clear (**dcble**)
- Data Cache Block Touch and Lock Set (**dcbtls**)
- Data Cache Block Touch for Store and Lock Set (**dcbtstls**)
- Instruction Cache Block Lock Clear (**icble**)
- Instruction Cache Block Touch and Lock Set (**icbtls**)

### 5.10.4 Programmable Page Characteristics

Cache and memory attributes are programmable on a per-page basis. In addition to the write-through, caching-inhibited, memory coherency enforced, and guarded characteristics defined by the WIMG bits, the endianness bit, E, allows selection of big- or little-endian byte ordering on a per-page basis.

In addition to the WIMGE bits, the MMU model defines user-definable page attribute bits U0–U3.

## 5.11 Core Complex Bus (CCB)

The core complex defines a versatile local bus interface that allows a wide range of system performance and system-complexity trade-offs. The interface defines the following buses:

- An address-out bus for mastering bus transactions
- An address-in bus for snooping internal resources
- Three tagged data buses

Two of the data buses are general-purpose data-in buses for reads, and the third is a data-out bus for writes. The two data-in buses feature support for out-of-order read transactions from two different sources simultaneously, and all three data buses may be operated concurrently. The address-in bus supports snooping for external management of the L1 caches and TLBs by other bus masters. The core complex broadcasts and snoops the cache and TLB management instructions accordingly. It is envisioned that a wide range of system implementations can be constructed from the defined interface.

## 5.12 Performance Monitoring

The e500 core provides a performance monitoring capability that allows counting of events such as processor clocks, instruction cache misses, data cache misses, mispredicted branches, and others. The count of these events may be configured to trigger a performance monitor exception following the e500 interrupt model. This interrupt is assigned to vector offset register IVOR35.

The register set associated with the performance monitoring function consists of counter registers, a global control register, and local control registers. These registers are read/write from supervisor mode, and each register is reflected to a corresponding read-only register for user mode. Two instructions, **mtpmr** and **mfpmr**, are provided for moving data to and from these registers. An overview of the performance monitoring registers is provided in the following sections.

### 5.12.1 Global Control Register

The PMGC0 register provides global control of the performance monitoring facility from supervisor mode. From this register all counters may be frozen, unfrozen, or configured to freeze on an enabled condition or event. Additionally, the performance monitoring facility may be disabled or enabled from this register. The contents of PMGC0 are reflected to UPMGC0, which may be read from user mode using the **mfpmr** instruction.

### 5.12.2 Performance Monitor Counter Registers

There are four counter registers (PCM0–PCM3) provided in the performance monitoring facility. These 32-bit registers hold the current count for software-selectable events and can be programmed to generate an exception on overflow. These registers may be written or read from supervisor mode using the **mtpmr** and **mfpmr** instructions. The contents of these registers are reflected to UPCM0–UPCM3, which can be read from user mode with **mfpmr**.

Performance monitor exceptions occur only if all of the following conditions are met:

- A counter is in the overflow state.
- The counter's overflow signaling is enabled.
- Overflow exception generation is enabled in PMGC0.
- MSR[EE] is set.



### 5.12.3 Local Control Registers

For each of the counter registers, there are two corresponding local control registers. These two registers specify which of the 128 available events is to be counted, what specific action is to be taken on overflow, and various options for freezing a counter value under given modes or conditions.

- PMLCa0–PMLCa3 provide fields that allow freezing of the corresponding counter in user mode, supervisor mode, or under software control. Additionally, the overflow condition may be enabled or disabled from this register. The contents of these registers are reflected to UPMLCa0–UPMLCa3, which can be read from user mode with **mfpmr**.
- PMLCb0–PMLCb3 provide count scaling for each counter register using configurable threshold and multiplier values. The threshold is a 6-bit value and the multiplier is a 3-bit encoded value, allowing eight multiplier values in the range of 1 to 128. Any counter may be configured to increment only when an event occurs more than [threshold × multiplier] times. The contents of these registers are reflected to UPMLCb0–UPMLCb3, which can be read from user mode with **mfpmr**.

## 5.13 Legacy Support of Power Architecture Technology

This section provides an overview of the architectural differences and compatibilities of the e500 core compared with the AIM Power Architecture technology. The two levels of the e500 programming environment are as follows:

- User level—This defines the base user-level instruction set, user-level registers, data types, memory conventions, and the memory and programming models seen by application programmers.
- Supervisor level—This defines supervisor-level resources typically required by an operating system, the memory management model, supervisor level registers, and the exception model.

Like all devices that implement the Power Architecture technology, in general, the e500 core supports the user-level architecture. The following sections are intended to highlight the main differences. For specific implementation details refer to the relevant chapter.

### 5.13.1 Instruction Set Compatibility

The following sections generally describe the user and supervisor instruction sets.

#### 5.13.1.1 User Instruction Set

The e500 core executes legacy user-mode binaries and object files except for the following:

- The e500 supports vector and scalar single-precision floating-point operations as part of the SPE. The e500v2 supports scalar double-precision floating-point instructions. These instructions have different encoding than the AIM definition of the architecture. Additionally, the e500 core uses GPRs for floating-point operations, rather than the FPRs defined by the UISA. Most porting of floating-point operations can be handled by recompiling.
- String instructions are not implemented on the e500; therefore, trap emulation must be provided to ensure backward compatibility.

### 5.13.1.2 Supervisor Instruction Set

The supervisor mode instruction set defined by the PowerPC architecture is compatible with the e500 with the following exceptions:

- The MMU architecture is different, so some TLB manipulation instructions have different semantics.
- Instructions that support the BATs and segment registers are not implemented.

### 5.13.2 Memory Subsystem

The architecture provides separate instruction and data memory resources. The e500 provides additional cache control features, including cache locking.

### 5.13.3 Exception Handling

Exception handling is generally the same as that defined in the AIM version of the architecture for the e500, with the following differences:

- The critical interrupt provides an extra level of interrupt nesting. The critical interrupt includes external critical and watchdog timer time-out inputs.
- The machine check exception uses the Return from Machine Check Interrupt instruction, **rfmci**, and two machine check save/restore registers, MCSRR0 and MCSRR1.
- IVPR and IVORs set interrupt vectors individually, but they can be set to the address offsets defined in the OEA to provide compatibility.
- The embedded category does not define a reset vector; execution begins at a fixed virtual address, 0xFFFF\_FFFC.
- Timer services are generally compatible, although the embedded category defines a new decremter auto reload feature, the fixed-interval timer critical interrupt, and the watchdog timer interrupt, which are implemented in the e500 core.

An overview of the interrupt and exception handling capabilities of the e500 core can be found in [Section 5.8, “Interrupts and Exception Handling.”](#)

### 5.13.4 Memory Management

The embedded category defines resources for fixed 4-Kbyte pages and multiple, variable page sizes that can be configured in a single implementation. TLB management is provided with new instructions and SPRs.

### 5.13.5 Reset

Embedded category–compliant cores do not share a common reset vector with the AIM version of the architecture. Instead, at reset fetching begins at address 0xFFFF\_FFFC. In addition, the Freescale MMU category defines specific aspects of the MMU page translation and protection mechanisms. Unlike the AIM version of the core, as soon as instruction fetching begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

MMU operations are described in the EREF.

### 5.13.6 Little-Endian Mode

Unlike the AIM version of the architecture, where little-endian mode is controlled on a system basis, the embedded category allows control of byte ordering on a memory page basis. In addition, the little-endian byte ordering used is true little endian.

## 5.14 PowerQUICC III Implementation Details

Table 5-8 summarizes details of the PowerQUICC III-specific implementation of the e500 core.

**Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation**

Feature	PowerQUICC III Implementation
Cache protocol	The L2 cache is a write-through cache and does not support MESI cache protocol.
Multiprocessor functionality	Because PowerQUICC III is designed for a uniprocessor environment, the following e500 functionality is not implemented: <ul style="list-style-type: none"> <li>The memory coherence bit, M, which is controlled through MAS2[M] and MAS4[MD] has no effect.</li> <li>HID1[ABE] has meaning only in that it must be set to ensure that cache and TLB management instructions operate properly with respect to the L2 cache.</li> <li>Dynamic snooping does not occur in power-stopped state (see the note below in the entry for dynamic bus snooping).</li> </ul>
Nexus support	Nexus is not supported. The Nexus processor ID register (NPIDR) and the Nexus bus enable bit (HID1[NEXEN]) are not supported.
R1 and R2 data bus parity	R1 and R2 data bus parity are disabled on PowerQUICC III devices. HID1[R1DPE,R2DPE] are reserved.
Dynamic bus snooping	The PowerQUICC III devices do not perform dynamic bus snooping as described here. That is, when the e500 core is in core-stopped state (which is the state of the core when the PowerQUICC III device is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, L1 caches should be flushed if coherency is required during these power-down modes. For more information, see <a href="#">Section 19.5.1.9, “Snooping in Power-Down Modes.”</a>
Supported TCR[WRC]	PowerQUICC III devices define values for 00, 01, 10, and 11, as described in <a href="#">Table 6-9 on page 6-15.</a>
SPE and floating-point categories	The SPE (which includes the embedded vector and scalar floating-point instructions) will not be implemented in the next generation of PowerQUICC devices. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses these instructions at the assembly level or that uses SPE or floating-point intrinsics will require rewriting for upward compatibility with next generation PowerQUICC devices. The e500v2 core implements SPE double-precision floating-point instructions. Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale Semiconductor will also provide future libraries to support next generation PowerQUICC devices.
HID0 implementation	SEL_TBCLK bit. Selects time base clock. If this bit is set and the time base is enabled, the time base is based on the TBCLK input, which on the PowerQUICC III devices is RTC.

**Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation (continued)**

Feature	PowerQUICC III Implementation
HID1 Implementation	<p>PLL_MODE. Set to 01</p> <p>PLL_CFG. This PowerQUICC III device supports the following:</p> <p>0001_00Ratio of 2:1</p> <p>0001_01Ratio of 5:2 (2.5:1)</p> <p>0001_10Ratio of 3:1</p> <p>0001_11Ratio of 7:2 (3.5:1)</p> <p>NEXEN, R1DPE, R2DPE, MPXTT, MSHARS, SSHAR, ATS, and MID are not implemented</p> <p>On PowerQUICC III devices, ABE must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.</p> <p>Please refer to the description of HID1[RFXE] in <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”</a></p> <p>If RFXE is 0, conditions that cause the assertion of <i>core_fault_in</i> cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured:</p> <ul style="list-style-type: none"> <li>• ECM mapping errors: EEER[LAE] must be set. See <a href="#">Section 8.2.1.6, “ECM Error Enable Register (EEER).”</a></li> <li>• L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. See <a href="#">Section 7.3.1.4, “L2 Error Registers.”</a></li> <li>• DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See <a href="#">Section 9.4.1, “Register Descriptions.”</a></li> <li>• PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated.</li> </ul> <p>Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that a parity error can generate an interrupt. See <a href="#">Section 14.3.1.11, “Transfer Error Check Disable Register (LTEDR),”</a> and <a href="#">Section 14.3.1.12, “Transfer Error Interrupt Enable Register (LTEIR).”</a></p>
PIR value	The PIR value is all zeros on PowerQUICC III devices.
PVR value	<p>The PVR reset value is 0x80nn_nnnn. See <a href="#">Table 5-1</a> for specific values.</p> <p>PVR[VERSION] = 0x80nn</p> <p>PVR[REVISION] = 0xnnnn</p>
SVR value	The SVR reset value is 0x80nn_nnnn. See <a href="#">Table 5-1</a> for specific values.
Alternate time base	The alternate time base defines a time base counter similar to the time base defined in architecture. It is intended to be used for measuring time in implementation defined intervals. It differs from the defined Time Base in that it is not writable and always counts up, wrapping when the 64-bit count overflows. It defines two SPRs, ATBL (SPR 526) and ATBL (SPR 527).

## Chapter 6

# Core Register Summary

This chapter describes the e500 register model and indicates whether each register is defined by the Power Architecture technology, by the Freescale embedded category implementation standards (EIS), or by the implementation. For the programmer, drawing this distinction indicates the degree to which code is portable among Freescale processors.

This chapter provides reference material—figures for each register and complete descriptions of register fields, including how the registers are accessed, reset values, and whether they can be accessed by user- and supervisor-level software. Detailed discussions of how these registers are used are provided in *EREF: A Reference for Freescale Book E and the e500 Core* and the *PowerPC™ e500 Core Family Reference Manual*.

Note that all registers described here are implemented in the hardware as part of the e500 core.

### 6.1 Overview

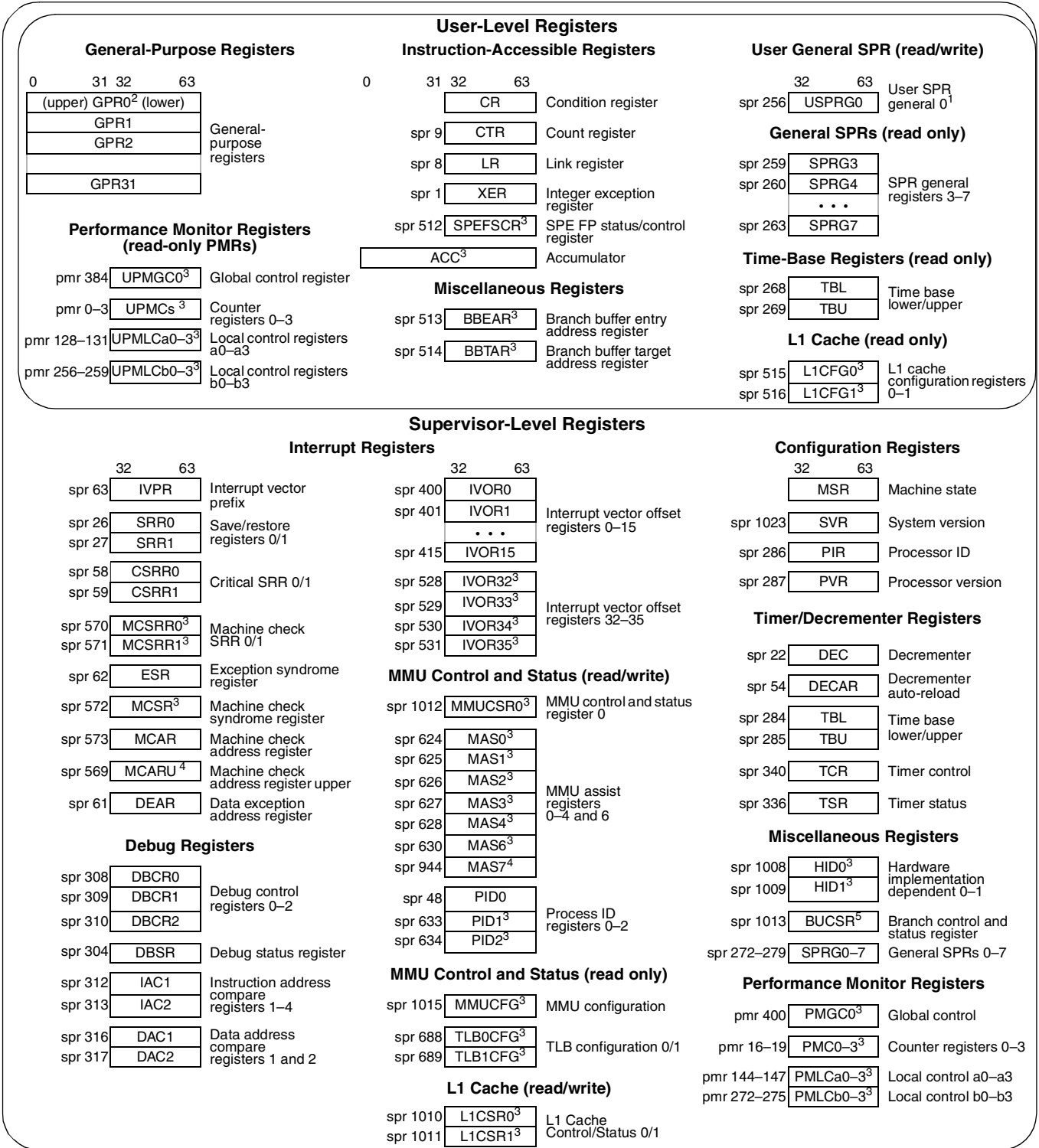
As shown in [Figure 6-1](#), most of the registers implemented are defined by the architecture, and most of those were defined by the AIM definition of the architecture and have changed very little. Additional registers and fields within registers are defined by the EIS and by the implementation.

The Power Architecture technology defines some register fields in a very general way, leaving some details as implementation specific. In some cases, this more specific functionality is defined by the EIS; in others it is left up to the processor. This chapter identifies the level at which each features is defined.

References to e500 are true for both the e500v1 and e500v2.

#### 6.1.1 Register Set

[Table 6-1](#) shows the e500 register set, grouped by whether they can be accessed by user- or supervisor-level software. Unless otherwise indicated, these registers are defined by the base or embedded category.



<sup>1</sup> USPRG0 is a separate physical register from SPRG0.  
<sup>2</sup> The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.  
<sup>3</sup> These registers are defined by the EIS and are not part of the Book E architecture.  
<sup>4</sup> e500v2 only  
<sup>5</sup> These registers are e500-specific

Figure 6-1. Core Register Model

## 6.2 Register Model for 32-Bit Implementations

Embedded 32-bit processors implement the following types of software-accessible registers:

- Architecture-defined registers that are accessed as part of instruction execution. These include the following:
  - Registers used for computation. These include the following:
    - General-purpose registers (GPRs)—The 32 GPRs hold source and destination operands for load, store, arithmetic, and computational instructions, and to read and write to other registers. The e500 implements these as 64-bit registers for use with 64-bit load, store, and merge instructions, as described in [Section 6.3.1, “General-Purpose Registers \(GPRs\).”](#)
    - Integer exception register (XER)—Bits in this register are set based on the operation of an instruction considered as a whole, not on intermediate results. (For example, the Subtract from Carrying instruction (**subfc**), the result of which is specified as the sum of three values, sets bits in the XER based on the entire operation, not on an intermediate sum.)  
These registers are described in [Section 6.3, “Registers for Computational Operations.”](#)
  - Condition register (CR)—Used to record conditions such as overflows and carries that occur as a result of executing arithmetic instructions (including those implemented by the SPE). The CR is described in [Section 6.4, “Registers for Branch Operations.”](#)
  - Machine state register (MSR)—Used by the operating system to configure parameters such as user/supervisor mode, address space, and enabling of asynchronous interrupts. This register is described in [Section 6.5.1, “Machine State Register \(MSR\),”](#) grouped with processor control SPRs.
- Special-purpose registers (SPRs) are accessed explicitly using **mtspr** and **mfspir** instructions. These registers are listed in [Table 6-1 in Section 6.2.1, “Special-Purpose Registers \(SPRs\).”](#)
- Freescale EIS– and e500-defined SPRs that are accessed explicitly using **mtspir** and **mfspir** are listed in [Table 6-2 in Section 6.2.1, “Special-Purpose Registers \(SPRs\).”](#)
- Freescale EIS–defined performance monitor registers (PMRs). These registers are similar to SPRs, but are accessed with Freescale EIS–defined move to and move from PMR instructions (**mtpmr** and **mfpmr**).

In this chapter, SPRs are grouped by function as follows:

- [Section 6.4, “Registers for Branch Operations,”](#) describes the count register (CTR) and the link register (LR).
- [Section 6.5, “Processor Control Registers”](#)
- [Section 6.6, “Timer Registers”](#)
- [Section 6.7, “Interrupt Registers”](#)
- [Section 6.8, “Software-Use SPRs \(SPRG0–SPRG7 and USPRG0\),”](#) describes SPRs defined for software use.
- [Section 6.9, “Branch Target Buffer \(BTB\) Registers,”](#) describes e500-specific registers defined to support the e500 tabs.
- [Section 6.10, “Hardware Implementation-Dependent Registers,”](#) describes HID0 and HID1.
- [Section 6.11, “L1 Cache Configuration Registers”](#)

- Section 6.12, “MMU Registers”
- Section 6.13, “Debug Registers”
- Section 6.14, “Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)”

The e500 core implements 64-bit GPRs, the upper 32 bits of which are used only with 64-bit load, store, and merge instructions.

## 6.2.1 Special-Purpose Registers (SPRs)

Table 6-1 summarizes SPRs. The SPR numbers are used in the instruction mnemonics. Bit 5 in an SPR number indicates whether an SPR is accessible from user- or supervisor-level software. An **mtspr** or **mfspr** instruction that specifies an unsupported SPR number is considered an invalid instruction.

In Table 6-1 and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- wlc indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

### NOTE

Writing to the following registers requires synchronization, as described in the “Synchronization Requirements” section in the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

- BTB locking registers—BBEAR, BBTAR, and BUCSR
- DBCR<sub>n</sub>
- HID<sub>n</sub>
- L1CSR<sub>n</sub>
- MMU registers—MAS<sub>n</sub>, MMUCSR0, PID<sub>n</sub>
- SPEFSCR

**Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation)**

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
CSRR0	Critical save/restore register 0	58	00001 11010	Read/Write	Yes	6.7.1.1/6-17
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-17
CTR	Count register	9	00000 01001	Read/Write	No	6.4.3/6-11



Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
DAC1	Data address compare 1	316	01001 11100	Read/Write	Yes	6.13.4/6-45
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0 <sup>1</sup>	308	01001 10100	Read/Write	Yes	6.13.1/6-39
DBCR1	Debug control register 1 <sup>1</sup>	309	01001 10101	Read/Write	Yes	
DBCR2	Debug control register 2 <sup>1</sup>	310	01001 10110	Read/Write	Yes	
DBSR	Debug status register	304	01001 10000	w1c <sup>2</sup>	Yes	6.13.2/6-43
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.7.1.5/6-18
DEC	Decrementer	22	00000 10110	Read/Write	Yes	6.6.4/6-16
DECAR	Decrementer auto-reload	54	00001 10110	Write only		
ESR	Exception syndrome register	62	00001 11110	Read/Write	Yes	6.7.1.8/6-19
IAC1	Instruction address compare 1	312	01001 11000	Read/Write	Yes	6.13.3/6-45
IAC2	Instruction address compare 2	313	01001 11001			
IVOR0	Critical input	400	01100 10000	Read/Write	Yes	6.7.1.7/6-18
IVOR1	Machine check interrupt offset	401	01100 10001			
IVOR2	Data storage interrupt offset	402	01100 10010			
IVOR3	Instruction storage interrupt offset	403	01100 10011			
IVOR4	External input interrupt offset	404	01100 10100			
IVOR5	Alignment interrupt offset	405	01100 10101			
IVOR6	Program interrupt offset	406	01100 10110			
IVOR8	System call interrupt offset	408	01100 11000			
IVOR10	Decrementer interrupt offset	410	01100 11010			
IVOR11	Fixed-interval timer interrupt offset	411	01100 11011			
IVOR12	Watchdog timer interrupt offset	412	01100 11100			
IVOR13	Data TLB error interrupt offset	413	01100 11101			
IVOR14	Instruction TLB error interrupt offset	414	01100 11110			
IVOR15	Debug interrupt offset	415	01100 11111			
IVPR	Interrupt vector	63	00001 11111			
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-11
PID	Process ID register <sup>3</sup>	48	00001 10000	Read/Write	Yes	6.12.1/6-32
PIR	Processor ID register	286	01000 11110	Read only	Yes	6.5.2/6-13
PVR	Processor version register	287	01000 11111	Read only	Yes	6.5.3/6-13

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page	
		Decimal	Binary				
SPRG0	SPR general 0	272	01000 10000	Read/Write	Yes	6.8/6-22	
SPRG1	SPR general 1	273	01000 10001	Read/Write	Yes		
SPRG2	SPR general 2	274	01000 10010	Read/Write	Yes		
SPRG3	SPR general 3	259	01000 00011	Read only	No <sup>4</sup>		
		275	01000 10011	Read/Write	Yes		
SPRG4	SPR general 4	260	01000 00100	Read only	No		
		276	01000 10100	Read/Write	Yes		
SPRG5	SPR general 5	261	01000 00101	Read only	No	6.8/6-22	
		277	01000 10101	Read/Write	Yes		
SPRG6	SPR general 6	262	01000 00110	Read only	No		
		278	01000 10110	Read/Write	Yes		
SPRG7	SPR general 7	263	01000 00111	Read only	No		
		279	01000 10111	Read/Write	Yes		
SRR0	Save/restore register 0	26	00000 11010	Read/Write	Yes	6.7.1.1/6-17	
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-17	
TBL	Time base lower	268	01000 01100	Read only	No	6.6.3/6-16	
		284	01000 11100	Write only	Yes		
TBU	Time base upper	269	01000 01101	Read only	No		
		285	01000 11101	Write only	Yes		
TCR	Timer control register	340	01010 10100	Read/Write	Yes		6.6.1/6-14
TSR	Timer status register	336	01010 10000	w1c <sup>5</sup>	Yes		6.6.2/6-15
USPRG0	User SPR general 0 <sup>6</sup>	256	01000 00000	Read/Write	No	6.8/6-22	
XER	Integer exception register	1	00000 00001	Read/Write	No	6.3.2/6-8	

<sup>1</sup> Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*

<sup>2</sup> The DBSR is read using **mf spr**. It cannot be directly written to. Instead, DBSR bits corresponding to 1 bits in the GPR can be cleared using **mt spr**.

<sup>3</sup> Implementations may support more than one PID. For implementations with multiple PIDs, the PID defined by the embedded category is PID0.

<sup>4</sup> User-mode read access to SPRG3 is implementation dependent.

<sup>5</sup> The TSR is read using **mf spr**. It cannot be directly written to. Instead, TSR bits corresponding to 1 bits in the GPR can be cleared using **mt spr**.

<sup>6</sup> USPRG0 is a separate physical register from SPRG0.

Table 6-2 describes the implementation-specific SPRs and SPRs defined by categories other than the base and embedded categories. Compilers should recognize the mnemonic names given in Table 6-2 when parsing instructions.

**Table 6-2. Additional SPRs (by SPR Abbreviation)**

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
BBEAR	Branch buffer entry address register <sup>1</sup>	513	Read/Write	No	6.9.1/6-23
BBTAR	Branch buffer target address register <sup>1</sup>	514	Read/Write	No	6.9.2/6-23
BUCSR	Branch unit control and status register <sup>1</sup>	1013	Read/Write	Yes	6.9.3/6-24
HID0	Hardware implementation dependent reg 0 <sup>1</sup>	1008	Read/Write	Yes	6.10.1/6-25
HID1	Hardware implementation dependent reg 1 <sup>1</sup>	1009	Read/Write	Yes	6.10.2/6-26
IVOR32	SPE unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-18
IVOR33	Floating-point data exception interrupt offset	529	Read/Write	Yes	
IVOR34	Floating-point round exception interrupt offset	530	Read/Write	Yes	
IVOR35	Performance monitor	531	Read/Write	Yes	
L1CFG0	L1 cache configuration register 0	515	Read only	No	6.11.3/6-30
L1CFG1	L1 cache configuration register 1	516	Read only	No	6.11.4/6-31
L1CSR0	L1 cache control and status register 0 <sup>1</sup>	1010	Read/Write	Yes	6.11.1/6-28
L1CSR1	L1 cache control and status register 1 <sup>1</sup>	1011	Read/Write	Yes	6.11.2/6-29
MAS0	MMU assist register 0 <sup>1</sup>	624	Read/Write	Yes	6.12.5.1/6-35
MAS1	MMU assist register 1 <sup>1</sup>	625	Read/Write	Yes	6.12.5.2/6-35
MAS2	MMU assist register 2 <sup>1</sup>	626	Read/Write	Yes	6.12.5.3/6-36
MAS3	MMU assist register 3 <sup>1</sup>	627	Read/Write	Yes	6.12.5.4/6-37
MAS4	MMU assist register 4 <sup>1</sup>	628	Read/Write	Yes	6.12.5.5/6-38
MAS6	MMU assist register 6 <sup>1</sup>	630	Read/Write	Yes	6.12.5.6/6-38
MAS7	MMU assist register 7 <sup>1</sup>	944	Read/Write	Yes	6.12.5.7/6-39
MCAR	Machine check address register	573	Read only	Yes	6.7.2.3/6-21
MCARU	Machine check address register upper	569	Read only	Yes	6.7.2.3/6-21
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.4/6-21
MCSRR0	Machine check save/restore register 0	570	Read/Write	Yes	6.7.2.1/6-20
MCSRR1	Machine check save/restore register 1	571	Read/Write	Yes	6.7.2.2/6-20
MMUCFG	MMU configuration register	1015	Read only	Yes	6.12.3/6-32
MMUCSR0	MMU control and status register 0 <sup>1</sup>	1012	Read/Write	Yes	6.12.2/6-32

**Table 6-2. Additional SPRs (by SPR Abbreviation) (continued)**

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
PID0	Process ID register 0 <sup>1</sup>	48	Read/Write	Yes	6.12.1/6-32
PID1	Process ID register 1 <sup>1</sup>	633	Read/Write	Yes	
PID2	Process ID register 2 <sup>1</sup>	634	Read/Write	Yes	
SPEFSCR	Signal processing and embedded floating-point status and control register <sup>1</sup>	512	Read/Write	No	6.14/6-45
SVR	System version register	1023	Read only	Yes	6.5.4/6-14
TLB0CFG	TLB configuration register 0	688	Read only	Yes	6.12.4/6-33
TLB1CFG	TLB configuration register 1	689	Read only	Yes	6.12.4.2/6-34

<sup>1</sup> Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

### 6.3 Registers for Computational Operations

The following sections describe general-purpose and integer exception registers.

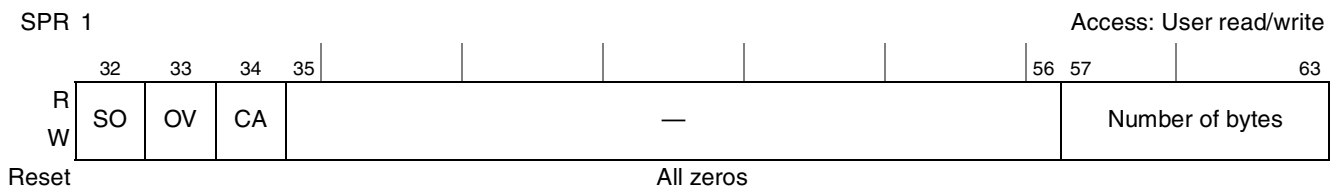
**NOTE**

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

#### 6.3.1 General-Purpose Registers (GPRs)

GPR0–GPR31 support integer operations. The instruction formats provide 5-bit fields for specifying the GPRs to be used in the execution of the instruction. Each GPR is a 64-bit register, although only 64-bit load, store, and merge instructions use GPR bits 0–31.

#### 6.3.2 Integer Exception Register (XER)



**Figure 6-2. Integer Exception Register (XER)**

Table 6-3. XER Field Description

Bits	Name	Description
32	SO	Summary overflow. Set when an instruction (except <b>mtspr</b> ) sets the overflow bit. Once set, SO remains set until it is cleared by <b>mtspr[XER]</b> or <b>mcrxr</b> . SO is not altered by compare instructions or by other instructions (except <b>mtspr[XER]</b> and <b>mcrxr</b> ) that cannot overflow. Executing <b>mtspr[XER]</b> , supplying the values 0 for SO and 1 for OV, causes SO to be cleared and OV to be set.
33	OV	Overflow. X-form add, subtract from, and negate instructions having OE = 1 set OV if the carry out of bit 32 is not equal to the carry out of bit 33, and clear OV otherwise to indicate a signed overflow. X-form multiply low word and divide word instructions having OE = 1 set OV if the result cannot be represented in 32 bits ( <b>mullwo</b> , <b>divwo</b> , and <b>divwuo</b> ) and clear OV otherwise. OV is not altered by compare instructions or by other instructions (except <b>mtspr[XER]</b> and <b>mcrxr</b> ) that cannot overflow.
34	CA	Carry. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA if there is a carry out of bit 32 and clear it otherwise. CA can be used to indicate unsigned overflow for add and subtract operations that set CA. Shift right algebraic word instructions set CA if any 1 bits are shifted out of a negative operand and clear CA otherwise. Compare instructions and instructions that cannot carry (except Shift Right Algebraic Word, <b>mtspr[XER]</b> , and <b>mcrxr</b> ) do not affect CA.
35–56	—	Reserved, should be cleared.
57–63	No. of bytes	Supports emulation of load and store string instructions. Specifies the number of bytes to be transferred by a load string indexed or store string indexed instruction.

## 6.4 Registers for Branch Operations

This section describes registers that support branch and CR operations.

### 6.4.1 Condition Register (CR)

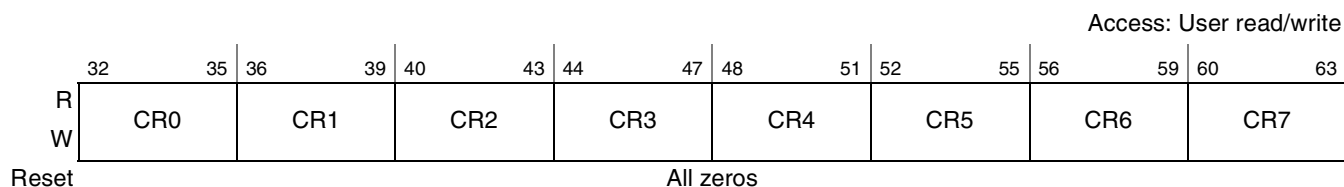


Figure 6-3. Condition Register (CR)

Table 6-4. BI Operand Settings for CR Fields

CRn Bits	CR Bits	BI	Description
CR0[0]	32	00000	Negative (LT)—Set when the result is negative. For SPE vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CR0[1]	33	00001	Positive (GT)—Set when the result is positive (and not zero). For SPE vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.
CR0[2]	34	00010	Zero (EQ)—Set when the result is zero. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.

Table 6-4. BI Operand Settings for CR Fields (continued)

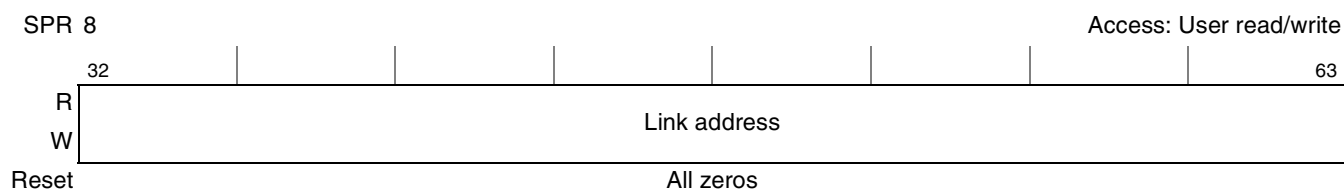
CRn Bits	CR Bits	BI	Description
CR0[3]	35	00011	Summary overflow (SO). Copy of XER[SO] at the instruction's completion. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CR1[0]	36	00100	Negative (LT) For SPE vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CR1[1]	37	00101	Positive (GT) For SPE vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.
CR1[2]	38	00110	Zero (EQ) For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR1[3]	39	00111	Summary overflow (SO) For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CRn[0]	40 44 48 52 56 60	01000 01100 10000 10100 11000 11100	Less than (LT) For integer compare instructions: rA < SIMM or rB (signed comparison) or rA < UIMM or rB (unsigned comparison). For SPE vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CRn[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than (GT) For integer compare instructions: rA > SIMM or rB (signed comparison) or rA > UIMM or rB (unsigned comparison). For SPE vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.
CRn[2]	42 46 50 54 58 62	01010 01110 10010 10110 11010 11110	Equal (EQ) For integer compare instructions: rA = SIMM, UIMM, or rB. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CRn[3]	43 47 51 55 59 63	01011 01111 10011 10111 11011 11111	Summary overflow (SO) For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.

The bits of CR0 are interpreted as described in [Table 6-5](#).

**Table 6-5. CR0 Bit Descriptions**

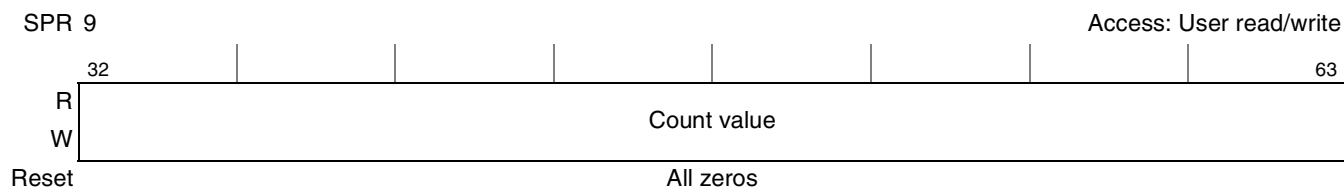
CR Bit	Name	Description
32	Negative (LT)	Bit 32 of the result is equal to 1.
33	Positive (GT)	Bit 32 of the result is equal to 0 and at least one bit from 33–63 of the result is non-zero.
34	Zero (EQ)	Bits 32–63 of the result are equal to 0.
35	Summary overflow (SO)	This is a copy of the final state of XER[SO] at the completion of the instruction.

## 6.4.2 Link Register (LR)



**Figure 6-4. Link Register (LR)**

## 6.4.3 Count Register (CTR)

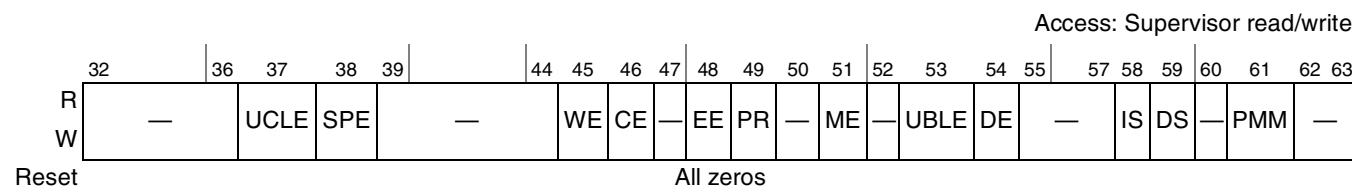


**Figure 6-5. Count Register (CTR)**

## 6.5 Processor Control Registers

This section addresses machine state, processor ID, and processor version registers.

### 6.5.1 Machine State Register (MSR)



**Figure 6-6. Machine State Register (MSR)**

Table 6-6. MSR Field Descriptions

Bits	Name	Description
32–36	—	Reserved, should be cleared. <sup>1</sup>
37	UCLE	User-mode cache lock enable. Used to restrict user-mode cache-line locking by the operating system 0 Any cache lock instruction executed in user-mode takes a cache-locking DSI exception and sets either ESR[DLK] or ESR[ILK]. This allows the operating system to manage and track the locking/unlocking of cache lines by user-mode tasks. 1 Cache-locking instructions can be executed in user-mode and they do not take a DSI for cache-locking (they may still take a DSI for access violations though).
38	SPE	SPE enable. (e500-specific). 0 If software attempts to execute an instruction that accesses the upper word of a GPR, the SPE unavailable exception is taken. 1 Software can execute the following instructions:  These instructions include the SPE instructions, embedded double-precision, and single-precision vector floating-point instructions. (That is, all instructions that access the upper half of the 64-bit GPRs.)
39–44	—	Reserved, should be cleared. <sup>1</sup>
45	WE	Wait state enable. Allows the core complex to signal a request for power management, according to the states of HID0[DOZE], HID0[NAP], and HID0[SLEEP]. 0 The processor is not in wait state and continues processing. No power management request is signaled to external logic. 1 The processor enters wait state by ceasing to execute instructions and entering low-power mode. Details of how wait state is entered and exited and how the processor behaves in the wait state are implementation-dependent. On the e500, MSR[WE] gates the DOZE, NAP, and SLEEP outputs from the core complex; as a result, these outputs negate to the external power management logic on entry to the interrupt and then return to their previous state on return from the interrupt. WE is cleared on entry to any interrupt and restored to its previous state upon return.
46	CE	Critical enable 0 Critical input and watchdog timer interrupts are disabled. 1 Critical input and watchdog timer interrupts are enabled.
47	—	Reserved, should be cleared. <sup>1</sup>
48	EE	External enable 0 External input, decremter, fixed-interval timer, and performance monitor interrupts are disabled. 1 External input, decremter, fixed-interval timer, and performance monitor interrupts are enabled.
49	PR	User mode (problem state) 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (for example, GPRs, SPRs, and the MSR). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource. PR also affects memory access control
50	—	Reserved, should be cleared. <sup>1</sup>
51	ME	Machine check enable 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
52	—	Reserved, should be cleared. <sup>1</sup>
53	UBLE	In the e500, it is the user BTB lock enable bit. 0 User-mode execution of the BTB lock instructions is disabled; privileged instruction exception taken instead. 1 User-mode execution of the BTB lock instructions for user mode is enabled.



Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
54	DE	Debug interrupt enable. See the description of the DBSR[UDE] in <a href="#">Section 6.13.2, “Debug Status Register (DBSR)”</a> . 0 Debug interrupts are disabled. 1 Debug interrupts are enabled if DBCR0[IDM] = 1.
55–57	—	Reserved, should be cleared. <sup>1</sup>
58	IS	Instruction address space 0 The processor directs all instruction fetches to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs all instruction fetches to address space 1 (TS = 1 in the relevant TLB entry).
59	DS	Data address space 0 The processor directs data memory accesses to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs data memory accesses to address space 1 (TS = 1 in the relevant TLB entry).
60	—	Reserved, should be cleared. <sup>1</sup>
61	PMM	Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during execution of the marked process. MSR[PR] and MSR[PMM] together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCax, the state for which monitoring is enabled, counting is enabled.
62–63	—	Preserved for OEA-defined RI and LE, respectively

<sup>1</sup> An MSR bit that is reserved may be altered by a return from interrupt instruction.

## 6.5.2 Processor ID Register (PIR)

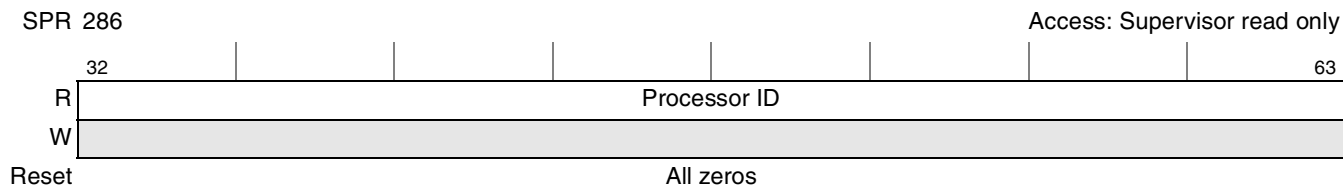


Figure 6-7. Processor ID Register (PIR)

## 6.5.3 Processor Version Register (PVR)

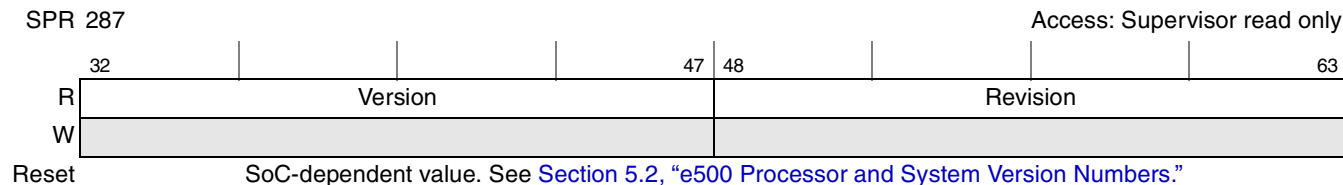
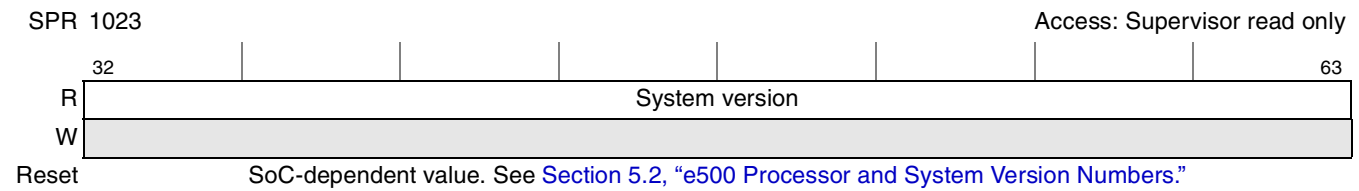


Figure 6-8. Processor Version Register (PVR)

**Table 6-7. PVR Field Descriptions**

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported. (See <a href="#">Section 5.2, “e500 Processor and System Version Numbers,”</a> for specific values.)
48–63	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level. (See <a href="#">Section 5.2, “e500 Processor and System Version Numbers,”</a> for specific values.)

## 6.5.4 System Version Register (SVR)



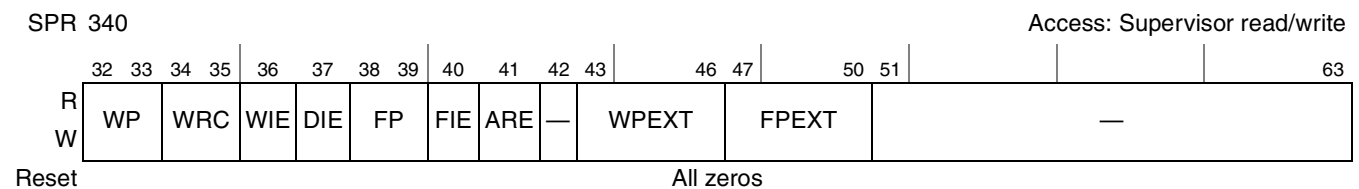
**Figure 6-9. System Version Register (SVR)**

**Table 6-8. SVR Field Descriptions**

Bits	Name	Description
32–63	System version	A 16-bit number that identifies the SoC version. See <a href="#">Section 5.2, “e500 Processor and System Version Numbers,”</a> for specific values.

## 6.6 Timer Registers

### 6.6.1 Timer Control Register (TCR)

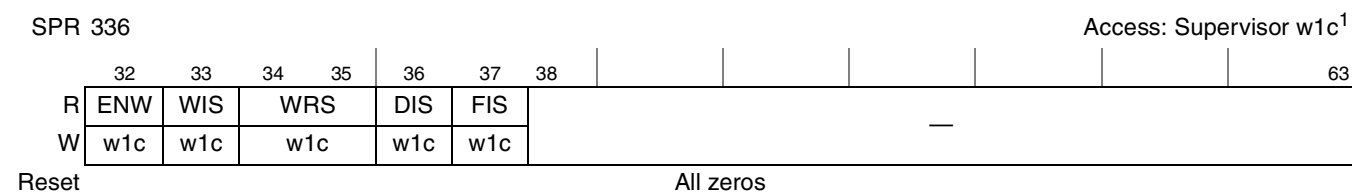


**Figure 6-10. Timer Control Register (TCR)**

Table 6-9. TCR Field Descriptions

Bits	Name	Description
32–33	WP	Watchdog timer period. When concatenated with WPEXT, specifies one of 64-bit locations of the time base used to signal a watchdog timer exception on a transition from 0 to 1. WPEXT[0–3]    WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) WPEXT[0–3]    WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
34–35	WRC	Watchdog timer reset control. This value is written into TSR[WRS] when a watchdog event occurs. WRC may be set by software but cannot be cleared by software, except by a software-induced reset. Once written to a non-zero value, WRC may no longer be altered by software. 00 No watchdog timer reset can occur. 01 Force processor checkstop on second timeout of watchdog timer 10 Assert processor reset output ( <i>core_hreset_req</i> ) on second timeout of watchdog timer 11 Reserved
36	WIE	Watchdog timer interrupt enable 0 Watchdog timer interrupts disabled 1 Watchdog timer interrupts enabled
37	DIE	Decrementer interrupt enable 0 Decrementer interrupts disabled 1 Decrementer interrupts enabled
38–39	FP	Fixed interval timer period. When concatenated with FPEXT, FP specifies one of 64 bit locations of the time base used to signal a fixed-interval timer exception on a transition from 0 to 1. FPEXT[0–3]    FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) FPEXT[0–3]    FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
40	FIE	Fixed interval interrupt enable 0 Fixed interval interrupts disabled 1 Fixed interval interrupts enabled
41	ARE	Auto-reload enable. Controls whether the DECAR value is reloaded into the DEC when the DEC value reaches 0000_0001. See <i>EREF: A Reference for Freescale Book E and the e500 Core</i> . 0 Auto-reload disabled 1 Auto-reload enabled
42	—	Reserved, should be cleared.
43–46	WPEXT	Watchdog timer period extension (see the description for WP)
47–50	FPEXT	Fixed-interval timer period extension (see the description for FP)
51–63	—	Reserved, should be cleared.

## 6.6.2 Timer Status Register (TSR)



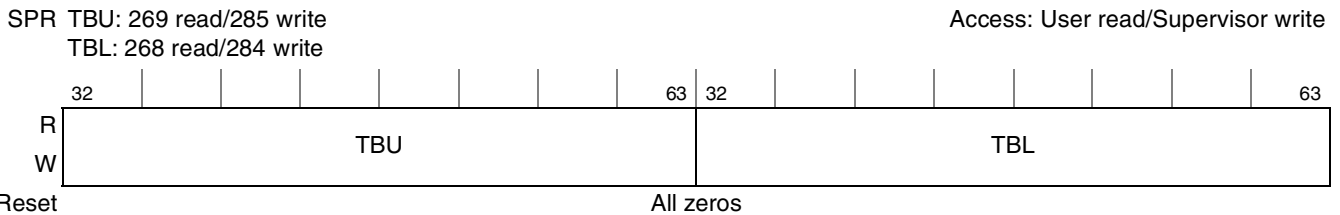
<sup>1</sup> Set by hardware. Read with **mfsp** and cleared with **mtspr** by writing ones to any TSR bit positions to be cleared and zeros in all other bit positions.

Figure 6-11. Timer Status Register (TSR)

**Table 6-10. TSR Field Descriptions**

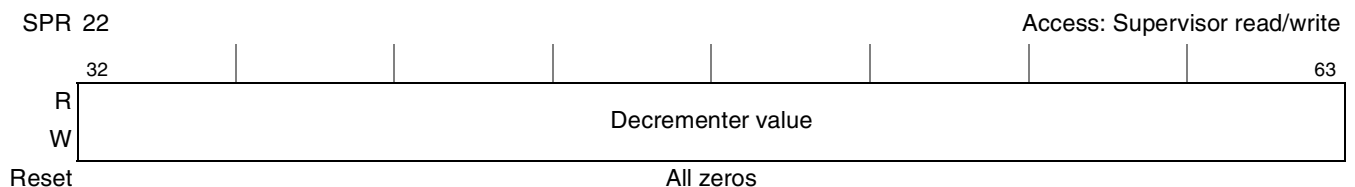
Bits	Name	Description
32	ENW	Enable next watchdog time. Functions as write-one-to-clear. 0 Action on next watchdog timer time-out is to set TSR[ENW] 1 Action on next watchdog timer time-out is governed by TSR[WIS] When a watchdog timer time-out occurs while WIS = 0 and the next watchdog time-out is enabled (ENW = 1), a watchdog timer exception is generated and logged by setting WIS. This is referred to as a watchdog timer first time out. A watchdog timer interrupt occurs if enabled by TCR[WIE] and MSR[CE]. To avoid another watchdog timer interrupt once MSR[CE] is reenabled, (assuming TCR[WIE] is not cleared instead), the interrupt handler must reset TSR[WIS].
33	WIS	Watchdog timer interrupt status. Functions as write-one-to-clear. 0 A watchdog timer event has not occurred. 1 A watchdog timer event occurred. When MSR[CE] = 1 and TCR[WIE] = 1, a watchdog timer interrupt is taken. See the description of ENW for more information about how WIS is used.
34–35	WRS	Watchdog timer reset status. Functions as write-one-to-clear. Defined at reset (value = 00). Set to TCR[WRC] when a reset is caused by the watchdog timer.
36	DIS	Decrementer interrupt status. Functions as write-one-to-clear. 0 A decrementer event has not occurred. 1 A decrementer event occurred. When MSR[EE] = TCR[DIE] = 1, a decrementer interrupt is taken.
37	FIS	Fixed-interval timer interrupt status. Functions as write-one-to-clear. 0 A fixed-interval timer event has not occurred. 1 A fixed-interval timer event occurred. When MSR[EE] = 1 and TCR[FIE] = 1, a fixed-interval timer interrupt is taken.
38–63	—	Reserved, should be cleared.

### 6.6.3 Time Base Registers



**Figure 6-12. Time Base Upper/Lower Registers (TBU/TBL)**

### 6.6.4 Decrementer Register



**Figure 6-13. Decrementer Register (DEC)**

## 6.6.5 Decrementer Auto-Reload Register (DECAR)

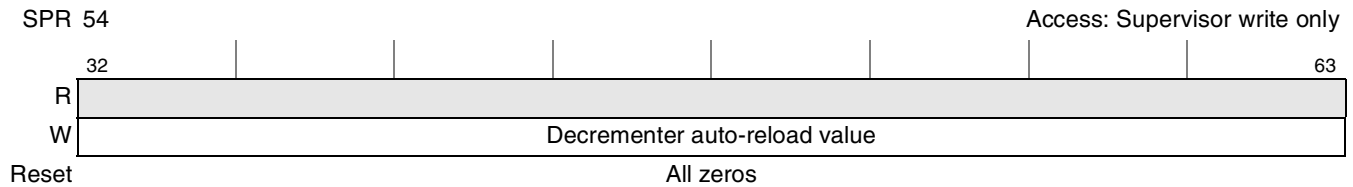


Figure 6-14. Decrementer Auto-Reload Register (DECAR)

## 6.7 Interrupt Registers

### 6.7.1 Interrupt Registers Defined by the Embedded and Base Categories

#### 6.7.1.1 Save/Restore Register 0 (SRR0)

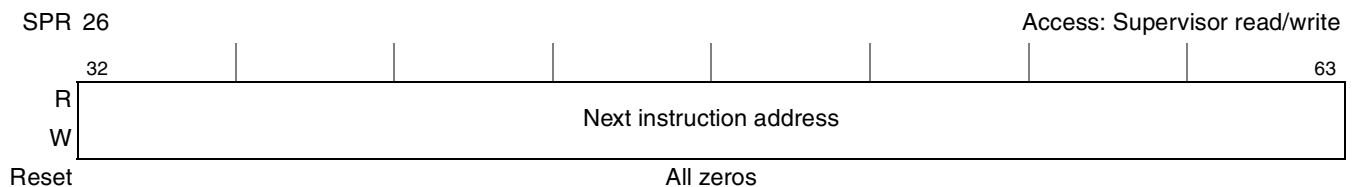


Figure 6-15. Save/Restore Register 0 (SRR0)

#### 6.7.1.2 Save/Restore Register 1 (SRR1)

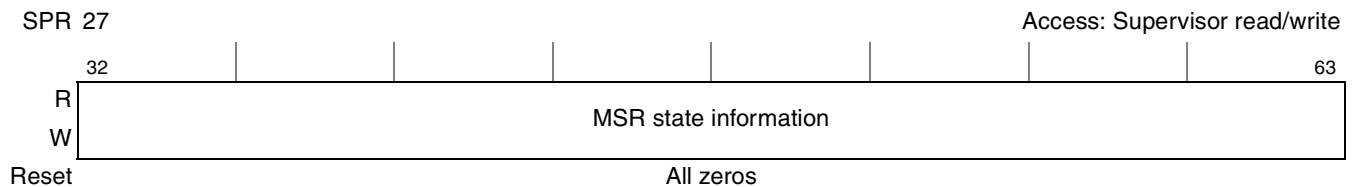


Figure 6-16. Save/Restore Register 1 (SRR1)

#### 6.7.1.3 Critical Save/Restore Register 0 (CSRR0)

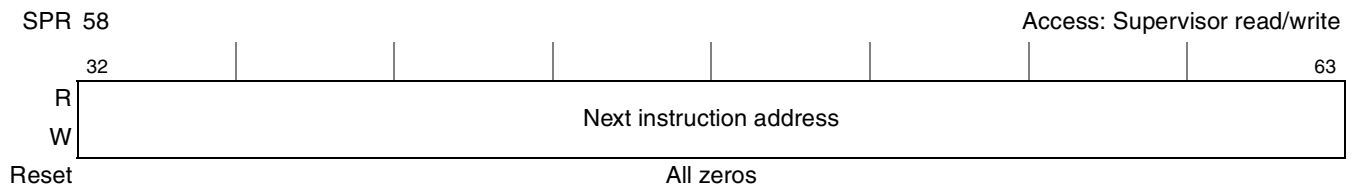


Figure 6-17. Critical Save/Restore Register 0 (CSRR0)

### 6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

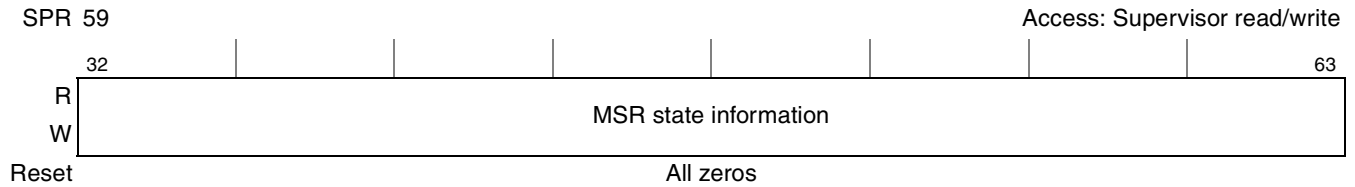


Figure 6-18. Critical Save/Restore Register 1 (CSRR1)

### 6.7.1.5 Data Exception Address Register (DEAR)

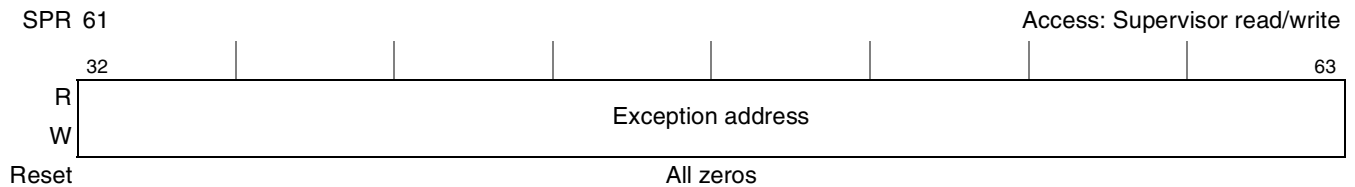


Figure 6-19. Data Exception Address Register (DEAR)

### 6.7.1.6 Interrupt Vector Prefix Register (IVPR)

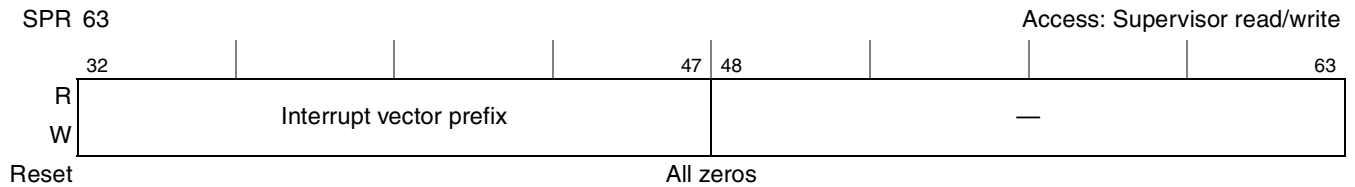


Figure 6-20. Interrupt Vector Prefix Register (IVPR)

### 6.7.1.7 Interrupt Vector Offset Registers (IVOR<sub>n</sub>)

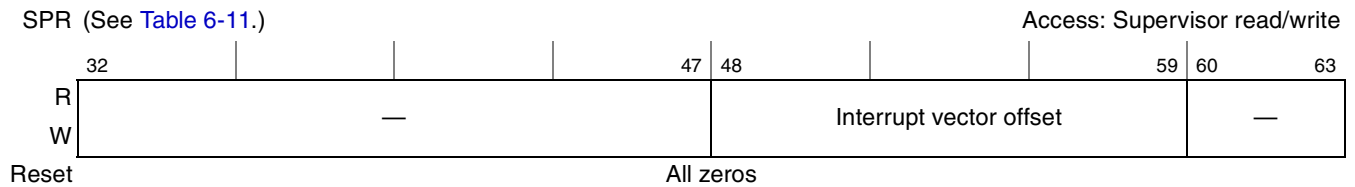


Figure 6-21. Interrupt Vector Offset Registers (IVOR<sub>n</sub>)

Table 6-11. IVOR Assignments

IVOR Number	SPR	Interrupt Type
IVOR0	400	Critical input
IVOR1	401	Machine check
IVOR2	402	Data storage
IVOR3	403	Instruction storage
IVOR4	404	External input

Table 6-11. IVOR Assignments (continued)

IVOR Number	SPR	Interrupt Type
IVOR5	405	Alignment
IVOR6	406	Program
IVOR8	408	System call
IVOR10	410	Decrementer
IVOR11	411	Fixed-interval timer interrupt
IVOR12	412	Watchdog timer interrupt
IVOR13	413	Data TLB error
IVOR14	414	Instruction TLB error
IVOR15	415	Debug
IVOR16–IVOR31	—	Reserved for future architectural use
IVOR32	528	SPE unavailable
IVOR33	529	Floating-point data exception
IVOR34	530	Floating-point round exception
IVOR35	531	Performance monitor
IVOR36–IVOR63	—	Allocated for implementation-dependent use

### 6.7.1.8 Exception Syndrome Register (ESR)

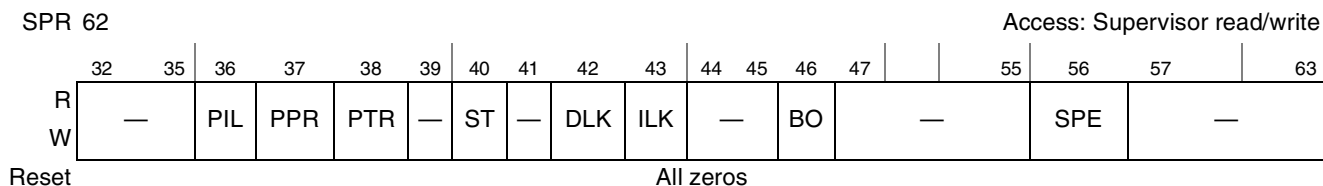


Figure 6-22. Exception Syndrome Register (ESR)

Table 6-12. ESR Field Descriptions

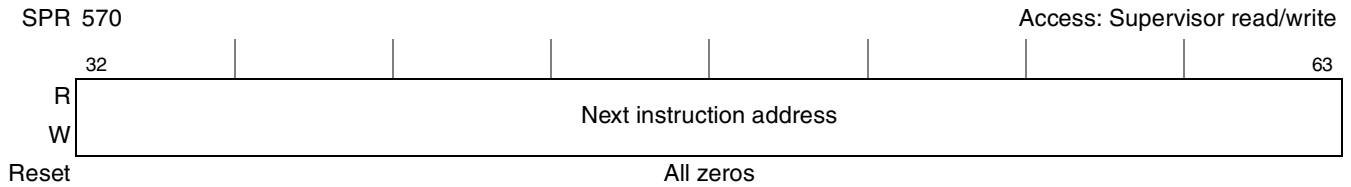
Bits	Name	Syndrome	Interrupt Types
32–35	—	Reserved, should be cleared.	—
36	PIL	Illegal instruction exception	Program
37	PPR	Privileged instruction exception	Program
38	PTR	Trap exception	Program
39	—	Reserved and permanently cleared because the e500 does not implement this FPU. Setting it has no effect.	—
40	ST	Store operation	Alignment, data storage, data TLB error

**Table 6-12. ESR Field Descriptions (continued)**

Bits	Name	Syndrome	Interrupt Types
41	—	Reserved, should be cleared.	—
42	DLK	Cache locking. Settings are implementation-dependent. 0 Default 1 On the e500, DLK is set when a DSI occurs because <b>dcbtls</b> , <b>dcbtstls</b> , or <b>dcblc</b> is executed in user mode while MSR[UCLC] = 0.	Data storage
43	ILK	Set when a DSI occurs because <b>icbtl</b> or <b>icblc</b> is executed in user mode (MSR[PR] = 1) and MSR[UCLC] = 0	Data storage
44–45	—	Reserved, should be cleared.	—
46	BO	Byte-ordering exception	Data storage, instruction storage
47–55	—	Reserved, should be cleared.	—
56	SPE	SPE exception bit (e500-specific) 0 Default 1 Any exception caused by an SPE or SPFP instruction	SPE unavailable
57–63	—	Reserved, should be cleared.	—

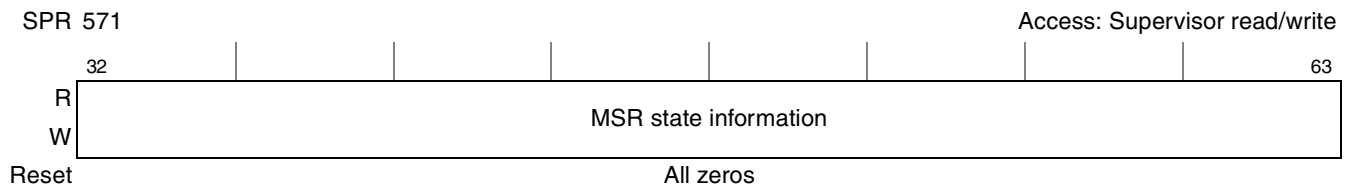
## 6.7.2 Additional Interrupt Registers

### 6.7.2.1 Machine Check Save/Restore Register 0 (MCSRR0)



**Figure 6-23. Machine Check Save/Restore Register 0 (MCSRR0)**

### 6.7.2.2 Machine Check Save/Restore Register 1 (MCSRR1)



**Figure 6-24. Machine Check Save/Restore Register 1 (MCSRR1)**



### 6.7.2.3 Machine Check Address Register (MCAR/MCARU)

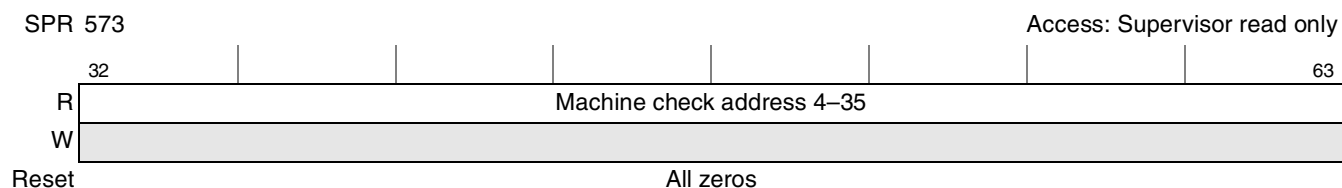


Figure 6-25. Machine Check Address Register (MCAR)



Figure 6-26. Machine Check Address Register Upper (MCARU)

### 6.7.2.4 Machine Check Syndrome Register (MCSR)

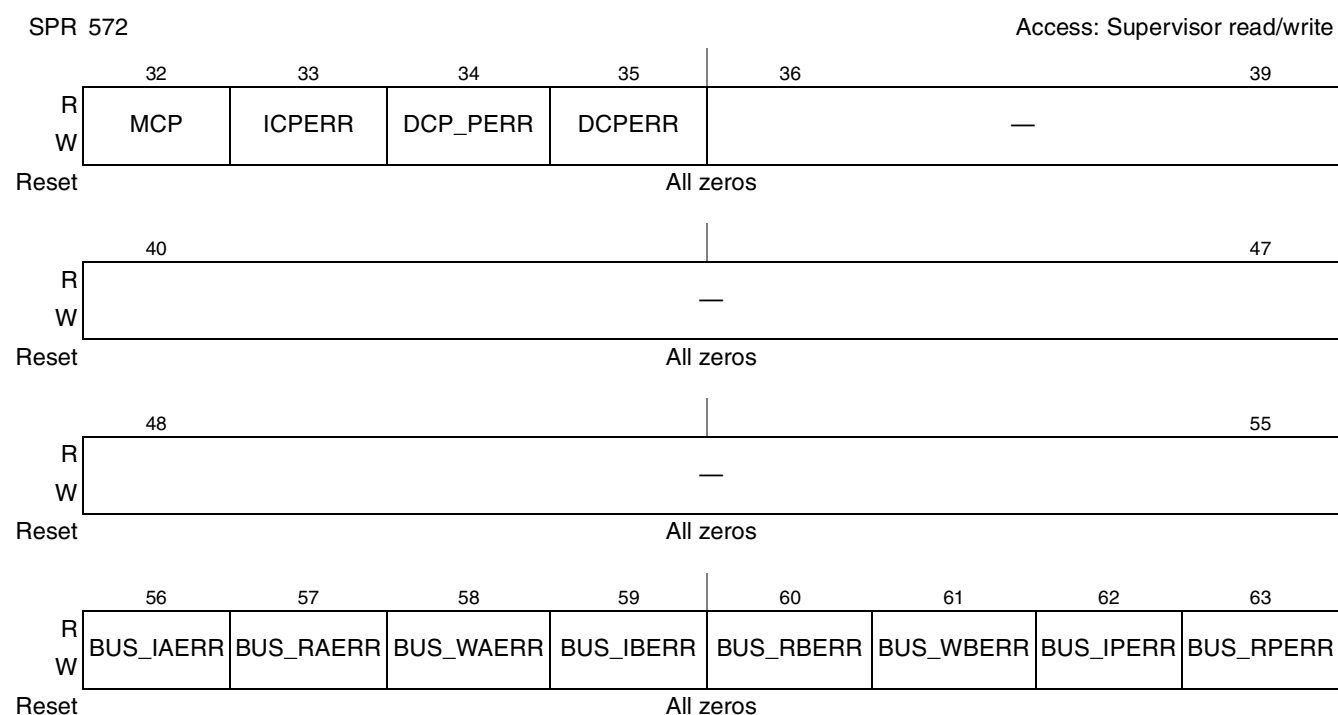


Figure 6-27. Machine Check Syndrome Register (MCSR)

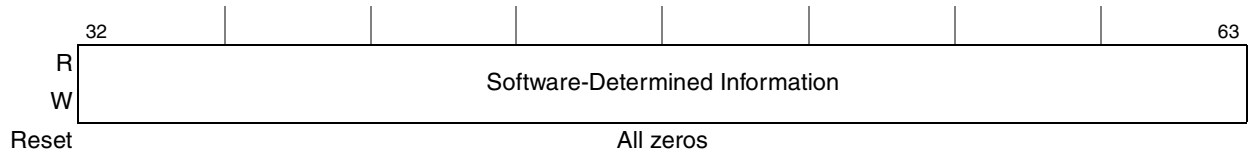
Table 6-13. MCSR Field Descriptions

Bits	Name	Description
32	MCP	Machine check input pin
33	ICPERR	Instruction cache parity error

**Table 6-13. MCSR Field Descriptions (continued)**

Bits	Name	Description
34	DCP_PERR	Data cache push parity error
35	DCPERR	Data cache parity error
36–55	—	Reserved, should be cleared.
56	BUS_IAERR	Bus instruction address error
57	BUS_RAERR	Bus read address error
58	BUS_WAERR	Bus write address error
59	BUS_IBERR	Bus instruction data bus error
60	BUS_RBERR	Bus read data bus error
61	BUS_WBERR	Bus write bus error
62	BUS_IPERR	Bus instruction parity error
63	BUS_RPERR	Bus read parity error

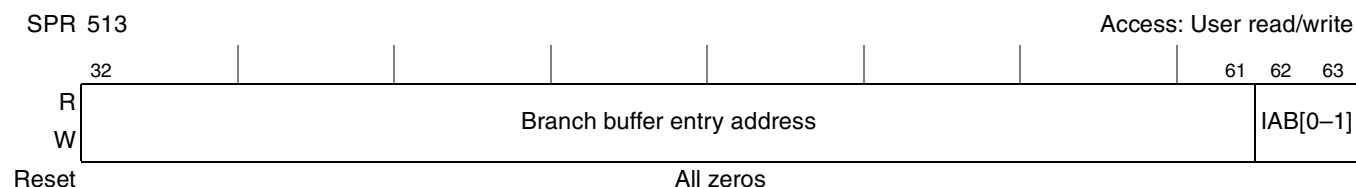
## 6.8 Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

SPR See [Table 6-14](#).Access: See [Table 6-14](#).**Figure 6-28. Software-Use SPRs (SPRG0–SPRG7 and USPRG0)****Table 6-14. SPR Assignments**

Name	SPR	Access	Level
SPRG0	272	Read/Write	Supervisor
SPRG1	273	Read/Write	Supervisor
SPRG2	274	Read/Write	Supervisor
SPRG3	259	Read only	User/Supervisor
	275	Read/Write	Supervisor
SPRG4	260	Read only	User/Supervisor
	276	Read/Write	Supervisor
SPRG5	261	Read only	User/Supervisor
	277	Read/Write	Supervisor
SPRG6	262	Read only	User/Supervisor
	278	Read/Write	Supervisor
SPRG7	263	Read only	User/Supervisor
	279	Read/Write	Supervisor
USPRG0	256	Read/Write	User/Supervisor

## 6.9 Branch Target Buffer (BTB) Registers

### 6.9.1 Branch Buffer Entry Address Register (BBEAR)

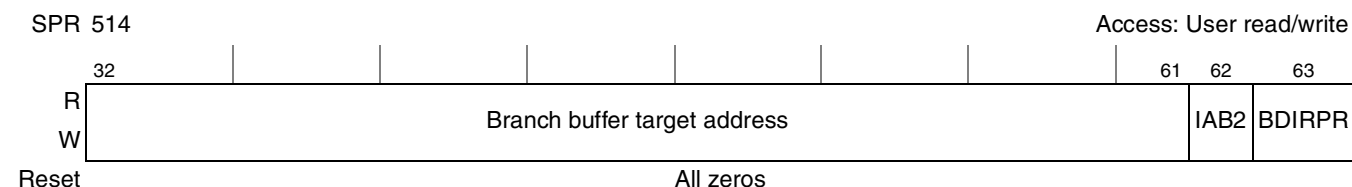


**Figure 6-29. Branch Buffer Entry Address Register (BBEAR)**

**Table 6-15. BBEAR Field Descriptions**

Bits	Name	Description
32–61	Branch buffer entry address	Branch buffer entry effective address bits 0–29
62–63	IAB[0–1]	Instruction after branch (with BBTAR[62]). 3-bit pointer that points to the instruction in the cache line after the branch. See the description in the <b>bblels</b> instruction in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.

### 6.9.2 Branch Buffer Target Address Register (BBTAR)

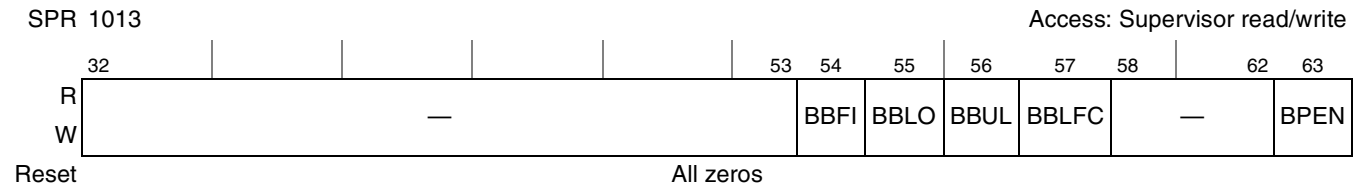


**Figure 6-30. Branch Buffer Target Address Register (BBTAR)**

**Table 6-16. BBTAR Field Descriptions**

Bits	Name	Description
32–61	Branch buffer target address	Branch buffer target effective address bits 0–29
62	IAB2	Instruction after branch bit 2 (with BBEAR[62–63]). IAB is a 3-bit pointer that points to the instruction in the cache line after the branch. See the description for <b>bblels</b> in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.
63	BDIRPR	Branch direction prediction. The user can pick the direction of the predicted branch. 0 The locked address is always predicted as not taken. 1 The locked address is always predicted as taken.

## 6.9.3 Branch Unit Control and Status Register (BUCSR)



**Figure 6-31. Branch Unit Control and Status Register (BUCSR)**

**Table 6-17. BUCSR Field Descriptions**

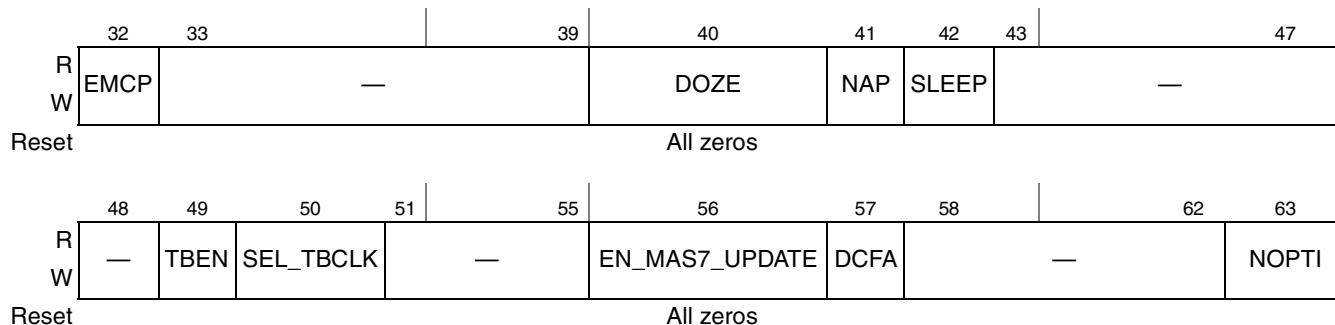
Bits	Name	Description
32–53	—	Reserved, should be cleared.
54	BBFI	Branch buffer flash invalidate. Clearing and then setting BBFI flash clears the valid bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBFI is always read as 0.
55	BBLO	Branch buffer lock overflow status 0 Indicates a lock overflow condition was not encountered in the branch buffer 1 Indicates a lock overflow condition was encountered in the branch buffer This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
56	BBUL	Branch buffer unable to lock 0 Indicates a lock overflow condition in the branch buffer 1 Indicates a lock set instruction failed in the branch buffer, for example, if the BTB is disabled. This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
57	BBLFC	Branch buffer lock bits flash clear. Clearing and then setting BBLFC flash clears the lock bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBLFC is always read as 0.
58–62	—	Reserved, should be cleared.
63	BPEN	Branch prediction enable 0 Branch prediction disabled 1 Branch prediction enabled (enables BTB to predict branches)

## 6.10 Hardware Implementation-Dependent Registers

### 6.10.1 Hardware Implementation-Dependent Register 0 (HID0)

SPR 1008

Access: Supervisor read/write



**Figure 6-32. Hardware Implementation-Dependent Register 0 (HID0)**

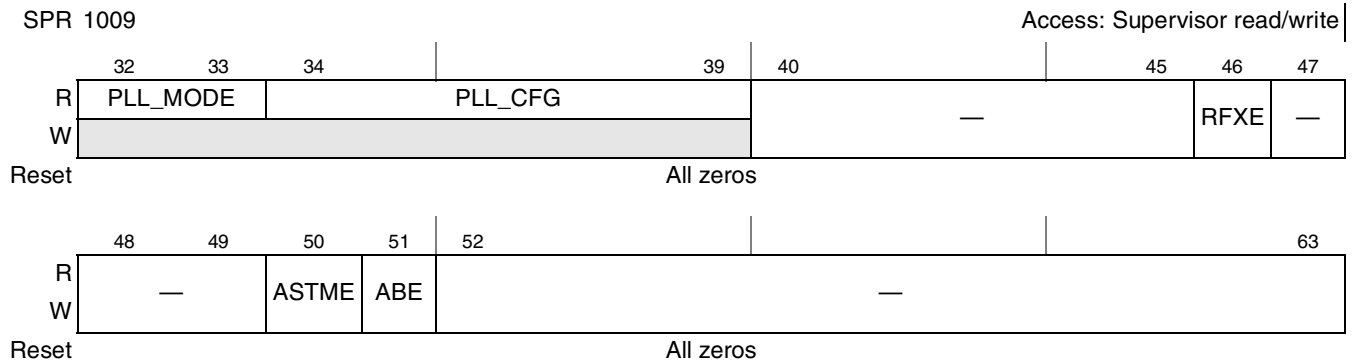
**Table 6-18. HID0 Field Descriptions**

Bits	Name	Description
32	EMCP	Enable machine check pin, $\overline{MCP}$ . Used to mask out further machine check exceptions caused by assertion of $\overline{MCP}$ . 0 $\overline{MCP}$ is disabled. 1 $\overline{MCP}$ is enabled. If MSR[ME] = 0, asserting $\overline{MCP}$ causes checkstop. If MSR[ME] = 1, asserting $\overline{MCP}$ causes a machine check exception.
33–39	—	Reserved, should be cleared.
40	DOZE	Doze power management mode. If MSR[WE] is set, this bit controls DOZE mode. 0 Core not in doze mode 1 Core in doze mode
41	NAP	Nap power management mode. If MSR[WE] is set, this bit controls NAP mode. 0 Core not in nap mode 1 Core in nap mode
42	SLEEP	Configure for sleep power management mode. Controls SLEEP mode if MSR[WE] is set. 0 Core not in sleep mode 1 Core in sleep mode
43–48	—	Reserved, should be cleared.
49	TBEN	Time base enable 0 Time base disabled (no counting) 1 Time base enabled <ul style="list-style-type: none"> <li>If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 0, the time base is updated every 8 bus clocks</li> <li>If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of <i>core_tclk</i> (sampled at bus rate). The maximum supported frequency can be found in the electrical specifications, but this value is approximately 25% of the bus clock frequency.</li> </ul>
50	SEL_TBCLK	Select time base clock. If the time base is enabled, this field functions as follows: 0 Time base is based on the processor clock 1 Time base is based on the TBCLK (RTC) input
51–55	—	Reserved, should be cleared.

**Table 6-18. HID0 Field Descriptions (continued)**

Bits	Name	Description
56	EN_MAS7_UPDATE	Enable MAS7 update (e500v2 only). Enables updating MAS7 by <b>tlbre</b> and <b>tlbsx</b> . 0 MAS7 is not updated by a <b>tlbre</b> or <b>tlbsx</b> . 1 MAS7 is updated by a <b>tlbre</b> or <b>tlbsx</b> .
57	DCFA	Data cache flush assist (e500v2 only). Force data cache to ignore invalid sets on miss replacement selection. 0 The data cache flush assist facility is disabled 1 The miss replacement algorithm ignores invalid entries and follows the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed load or <b>dcbz</b> instructions to eight per set. The bit should be set just before beginning a cache flush routine and should be cleared when the series of instructions is complete.
58–62	—	Reserved, should be cleared.
63	NOPTI	No-op the data and instruction cache touch instructions. 0 <b>dcbt</b> , <b>dcbstst</b> , and <b>icbt</b> are enabled. On the e500, if CT = 0, <b>icbt</b> is always a no-op, regardless of the value of NOPTI. If CT = 1, <b>icbt</b> does a touch load to an L2 cache, if one is present. 1 <b>dcbt</b> , <b>dcbstst</b> , and <b>icbt</b> are treated as no-ops; <b>dcblc</b> and <b>dcblts</b> are not.

## 6.10.2 Hardware Implementation-Dependent Register 1 (HID1)



**Figure 6-33. Hardware Implementation-Dependent Register 1 (HID1)**

**Table 6-19. HID1 Field Descriptions**

Bits	Name	Description
32–33	PLL_MODE	Read-only for integrated devices. 01 Fixed value for MPC8533E
34–39	PLL_CFG	Reflected directly from configuration input pins (read-only). PLL_CFG[0–4] corresponds to the integer divide ratio and PLL_CFG5 is the half-mode bit. The following values are supported: 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) Note that this value is also reflected to PORPLLSR[e500_Ratio]. See <a href="#">Section 19.4.1.1, “POR PLL Status Register (PORPLLSR).”</a>

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
40–45	—	Reserved, should be cleared.
46	RFXE	<p>Read fault exception enable. Enables the core to internally generate a machine check interrupt when <code>core_fault_in</code> is asserted. Depending on the value of MSR[ME], this results in either a machine check interrupt or a checkstop.</p> <p>0 Assertion of <code>core_fault_in</code> cannot cause a machine check. The core does not execute any instructions from a faulty instruction fetch and does not execute any load instructions that get their data from a faulty data fetch.</p> <p>On the e500v2, if these instructions are eventually required by the sequential programming model (that is, they are not in a speculative execution path), the e500v2 stalls until an asynchronous interrupt is taken. The e500v1 does not stall when faulty instructions or data are received, as described in the following note.</p> <p><b>Note:</b> The e500v1 does not stall when faulty instructions or data are received. Instead, it continues processing with faulty instructions or data. The only reliable way to prevent such behavior is to set RFXE, which causes a machine check before the faulty instructions or data are used. To avoid the use of faulty instructions or data and to have good error determination, software must set RFXE and program the PIC to interrupt the processor when errors occur. As a result, software must deal with multiple interrupts for the same fundamental problem.</p> <p>1 Assertion of <code>core_fault_in</code> causes a machine check if MSR[ME] = 1 or a checkstop if MSR[ME] = 0. The <code>core_fault_in</code> signal is asserted to the core when logic outside of the core has a problem delivering good data to the core. For example, the front-side L2 cache asserts <code>core_fault_in</code> when an ECC error occurs and ECC is enabled. As a second example, it is asserted when there is a master abort on a PCI transaction. See “Proper Reporting of Bus Faults” in the core complex bus chapter of the <i>PowerPC™ e500 Core Family Reference Manual</i>.</p> <p>The RFXE bit provides flexibility in error recovery. Typically, devices outside the core have some way other than the assertion of <code>core_fault_in</code> to signal the core that an error occurred. Usually, this is done by channeling interrupt requests through a programmable interrupt controller (PIC) to the core. In these cases, the assertion of <code>core_fault_in</code> is used only to prevent the core from using bad data before receiving an interrupt from the PIC (for example, an external or critical input interrupt). Possible combinations of RFXE and PIC configuration are as follows:</p> <ul style="list-style-type: none"> <li>• RFXE = 0 and the PIC is configured to interrupt the processor. In this configuration, the assertion of <code>core_fault_in</code> does not trigger a machine check interrupt. The core does not use the faulty instructions or data and may stall. The PIC interrupts the core so that error recovery can begin. This configuration allows the core to query the PIC and the rest of the system for more information about the cause of the interrupt, and generally provides the best error recovery capabilities.</li> <li>• RFXE = 1 and the PIC is not configured to interrupt the processor. This configuration provides quick error detection without the overhead of configuring the PIC. When the PIC is not configured, setting RFXE avoids stalling the core when <code>core_fault_in</code> is asserted. Determination of the root cause of the problem may be somewhat more difficult than it would be if the PIC were enabled.</li> <li>• RFXE = 1 and the PIC is configured to interrupt the processor. In this configuration, the core may receive two interrupts for the same fundamental error. The two interrupts may occur in any order, which may complicate error handling. Therefore, this is usually not an interesting configuration for a single-core device. This may, however, be an interesting configuration for multi-core devices in which the PIC may steer interrupts to a processor other than the one that attempted to fetch the faulty data.</li> <li>• RFXE = 0 and the PIC is not configured to interrupt the processor. This is not a recommended configuration. The processor may stall indefinitely due to an unreported error.</li> </ul>
47–49	—	Reserved, should be cleared.
50	ASTME	<p>Address bus streaming mode enable. This bit, along with the ECM stream control bits in the EEBACR, enables address bus streaming on the CCB. See <a href="#">Section 8.2.1.1, “ECM CCB Address Configuration Register (EEBACR)”</a></p> <p>0 Address bus streaming mode disabled</p> <p>1 Address bus streaming mode enabled</p>

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
51	ABE	Address broadcast enable. The e500 broadcasts cache management instructions ( <b>dcbst</b> , <b>dcblc</b> (CT = 1), <b>icblc</b> (CT = 1), <b>dcbf</b> , <b>dcbi</b> , <b>mbar</b> , <b>msync</b> , <b>tlbsync</b> , <b>icbi</b> ) based on ABE. ABE must be set to allow management of external L2 caches. 0 Address broadcasting disabled 1 Address broadcasting enabled
52–63	—	Reserved, should be cleared.

## 6.11 L1 Cache Configuration Registers

### 6.11.1 L1 Cache Control and Status Register 0 (L1CSR0)

SPR 1010

Access: Supervisor read/write

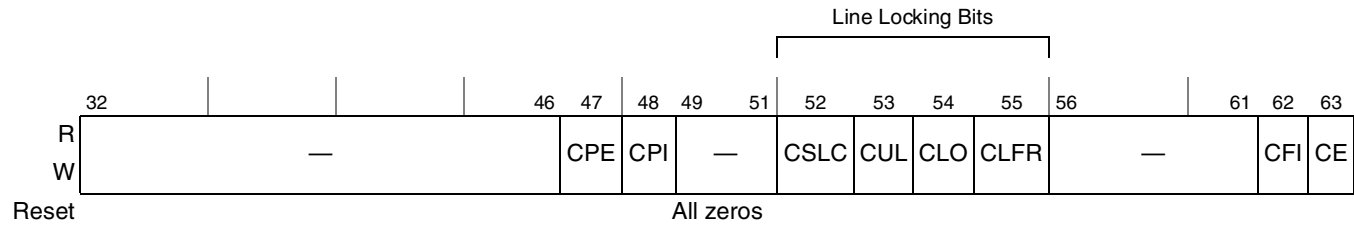


Figure 6-34. L1 Cache Control and Status Register 0 (L1CSR0)

Table 6-20. L1CSR0 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	CPE	(Data) Cache parity enable 0 Parity checking of the cache disabled 1 Parity checking of the cache enabled
48	CPI	(Data) Parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Cache parity must also be enabled (CPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	CSLC	(Data) Cache snoop lock clear. Sticky bit set by hardware if a <b>dcbi</b> snoop (either internally or externally generated) invalidated a locked cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can be cleared only by software. 0 The cache has not encountered a <b>dcbi</b> snoop that invalidated a locked line. 1 The cache has encountered a <b>dcbi</b> snoop that invalidated a locked line.
53	CUL	(Data) Cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the cache 1 Indicates a lock set instruction was not effective in the cache
54	CLO	(Data) Cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the cache 1 Indicates a lock overflow condition was encountered in the cache



Table 6-20. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
55	CLFR	(Data) Cache lock bits flash reset. Writing a 1 during a flash clear operation causes an undefined operation. Writing a 0 during a flash clear operation is ignored. Clearing occurs regardless of the enable (CE) value. 0 Default. 1 Hardware initiates a cache lock bits flash clear operation. This bit is cleared when the operation is complete.
56–61	—	Reserved, should be cleared.
62	CFI	(Data) Cache flash invalidate. 0 No cache invalidate. Writing a 0 to CFI during an invalidation operation is ignored. 1 Cache invalidation operation. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation. Invalidation occurs regardless of the enable (CE) value.
63	CE	(Data) Cache enable 0 The cache is neither accessed or updated. 1 Enables cache operation

## 6.11.2 L1 Cache Control and Status Register 1 (L1CSR1)

SPR 1011

Access: Supervisor read/write

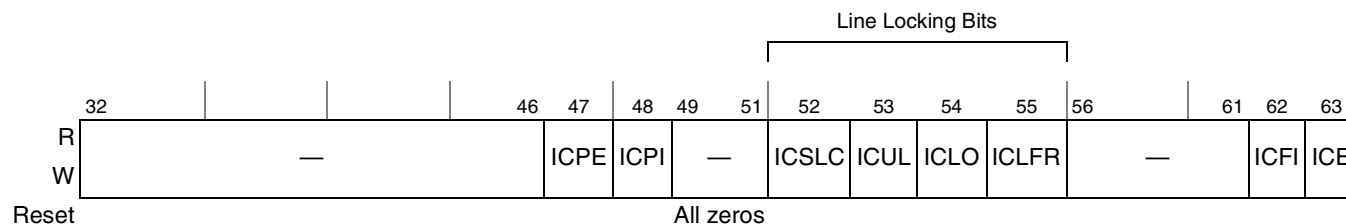


Figure 6-35. L1 Cache Control and Status Register 1 (L1CSR1)

Table 6-21. L1CSR1 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	ICPE	Instruction cache parity enable 0 Parity checking of the instruction cache disabled 1 Parity checking of the instruction cache enabled
48	ICPI	Instruction parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Note that instruction cache parity must also be enabled (ICPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	ICSLC	Instruction cache snoop lock clear. Sticky bit set by hardware if an <b>icbi</b> snoop (either internally or externally generated) invalidated a locked line in the instruction cache. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can only be cleared by software. 0 The instruction cache has not encountered an <b>icbi</b> snoop that invalidated a locked line. 1 The instruction cache has encountered an <b>icbi</b> snoop that invalidated a locked line.

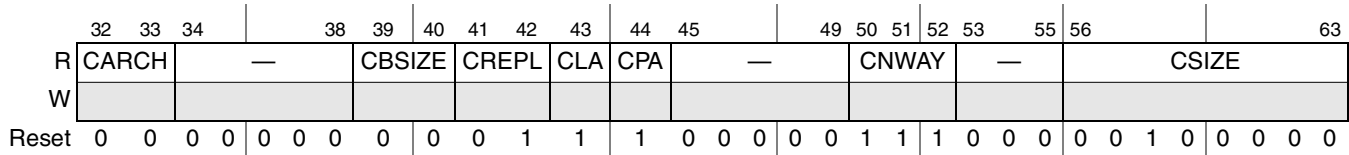
**Table 6-21. L1CSR1 Field Descriptions (continued)**

Bits	Name	Description
53	ICUL	Instruction cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the instruction cache 1 Indicates a lock set instruction was not effective in the instruction cache
54	ICLO	Instruction cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the instruction cache 1 Indicates a lock overflow condition was encountered in the instruction cache
55	ICLFR	Instruction cache lock bits flash reset. Writing 0 and then 1 flash clears the lock bit of all entries in the instruction cache; clearing occurs independently from the value of the enable bit (ICE). ICLFR is always read as 0.
56–61	—	Reserved, should be cleared.
62	ICFI	Instruction cache flash invalidate. Written to 0 and then 1 to flash clear the valid bit of all entries in the instruction cache; operates independently from the value of the enable bit (ICE). ICFI is always read as 0.
63	ICE	Instruction cache enable 0 The instruction cache is neither accessed or updated. 1 Enables instruction cache operation

### 6.11.3 L1 Cache Configuration Register 0 (L1CFG0)

SPR 515

Access: User read only



**Figure 6-36. L1 Cache Configuration Register 0 (L1CFG0)**

**Table 6-22. L1CFG0 Field Descriptions**

Bits	Name	Description
32–33	CARCH	Cache architecture 00 Harvard 01 Unified
34–38	—	Reserved, should be cleared.
39–40	CBSIZE	Cache line size 0 32 bytes 1 64 bytes
41–42	CREPL	Cache replacement policy 0 True LRU 1 Pseudo LRU
43	CLA	Cache locking available 0 Unavailable 1 Available

Table 6-22. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
44	CPA	Cache parity available 0 Unavailable 1 Available
45–49	—	Reserved, should be cleared.
50–52	CNWAY	Cache number of ways 111 Indicates 8 ways
53–55	—	Reserved, should be cleared.
56–63	CSIZE	Cache size 0x20 indicates 32 Kbytes

### 6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

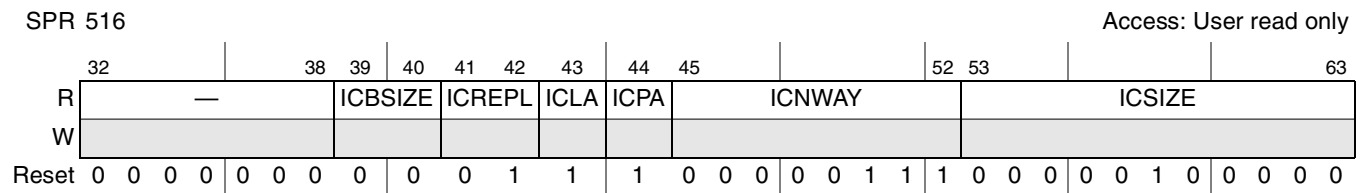


Figure 6-37. L1 Cache Configuration Register 1 (L1CFG1)

Table 6-23. L1CFG1 Field Descriptions

Bits	Name	Description
32–38	—	Reserved, should be cleared.
39–40	ICBSIZ	Instruction cache block size 00 Indicates block size of 32 bytes
41–42	ICREPL	Instruction cache replacement policy 01 Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available 1 Indicates available
44	ICPA	Instruction cache parity available 1 Indicates available
45–52	ICNWAY	Instruction cache number of ways 111 Indicates 8 ways
53–63	ICSIZE	Instruction cache size 0x20 indicates 32 Kbytes

## 6.12 MMU Registers

### 6.12.1 Process ID Registers (PID0–PID2)

SPR 48 (PID0) Access: Supervisor read/write  
 SPR 633 (PID1)  
 SPR 634 (PID2)

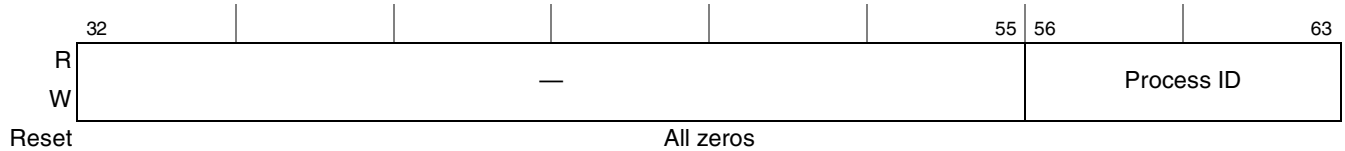


Figure 6-38. Process ID Registers (PID0–PID2)

### 6.12.2 MMU Control and Status Register 0 (MMUCSR0)

SPR 1012 Access: Supervisor read/write

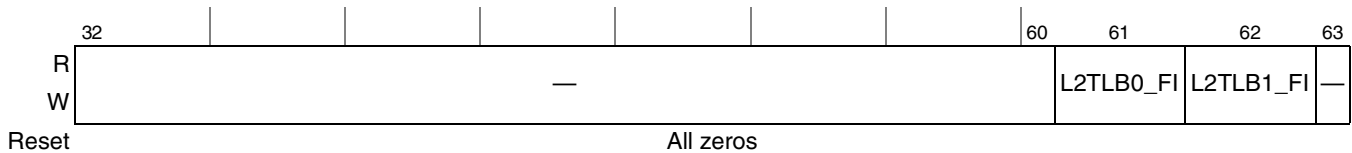


Figure 6-39. MMU Control and Status Register 0 (MMUCSR0)

Table 6-24. MMUCSR0 Field Descriptions

Bits	Name	Description
32–60	—	Reserved, should be cleared.
61	L2TLB0_FI	TLB0 flash invalidate (write to 1 to invalidate)
62	L2TLB1_FI	TLB1 flash invalidate (write 1 to invalidate) 0 No flash invalidate. Writing a 0 to this bit during an invalidation operation is ignored. 1 TLB1 invalidation operation. Hardware initiates a TLB1 invalidation operation. When this operation is complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation.
63	—	Reserved, should be cleared.

### 6.12.3 MMU Configuration Register (MMUCFG)

SPR 1015 Access: Supervisor read only

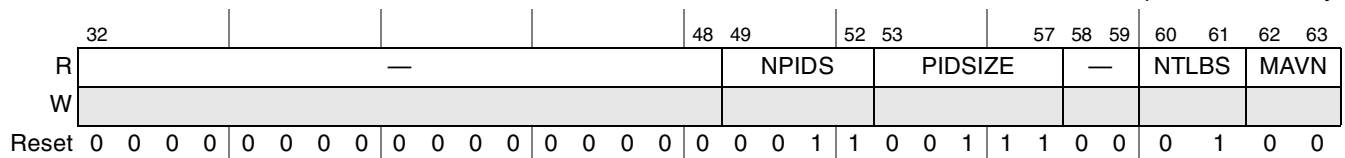


Figure 6-40. MMU Configuration Register (MMUCFG)

Table 6-25. MMUCFG Field Descriptions

Bits	Name	Description
32–48	—	Reserved, should be cleared.
49–52	NPIDS	Number of PID registers, a 4-bit field that indicates the number of PID registers provided by the processor. The e500 implements three PIDs.
53–57	PIDSIZE	PID register size. The 5-bit value of PIDSIZE is one less than the number of bits in each of the PID registers implemented by the processor. The processor implements only the least significant PIDSIZE+1 bits in the PID registers. 00111 indicates 8-bit registers. This is the value presented by the e500.
58–59	—	Reserved, should be cleared.
60–61	NTLBS	Number of TLBs. The value of NTLBS is one less than the number of software-accessible TLB structures that are implemented by the processor. NTLBS is set to one less than the number of TLB structures so that its value matches the maximum value of MAS0[TLBSEL]. 00 1 TLB 01 2 TLBs. This is the value presented by the e500. 10 3 TLBs 11 4 TLBs
62–63	MAVN	MMU architecture version number. Indicates the version number of the architecture of the MMU implemented by the processor. 0b00 indicates version 1.0.

## 6.12.4 TLB Configuration Registers (TLB<sub>n</sub>CFG)

### 6.12.4.1 TLB0 Configuration Register 0 (TLB0CFG)

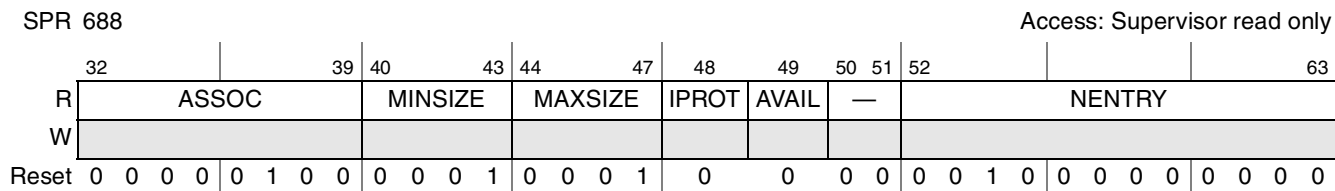


Figure 6-41. TLB Configuration Register 0 (TLB0CFG)

Table 6-26. TLB0CFG Field Descriptions

Bits	Name	Description
32–39	ASSOC	Associativity of TLB0 0x02 indicates associativity is 2-way set associative
40–43	MINSIZE	Minimum page size of TLB0 0x1 indicates smallest page size is 4K
44–47	MAXSIZE	Maximum page size of TLB0 0x1 indicates maximum page size is 4K
48	IPROT	Invalidate protect capability of TLB0 0 Indicates invalidate protection capability not supported
49	AVAIL	Page size availability of TLB0 0 No variable-sized pages available (MINSIZE = MAXSIZE)

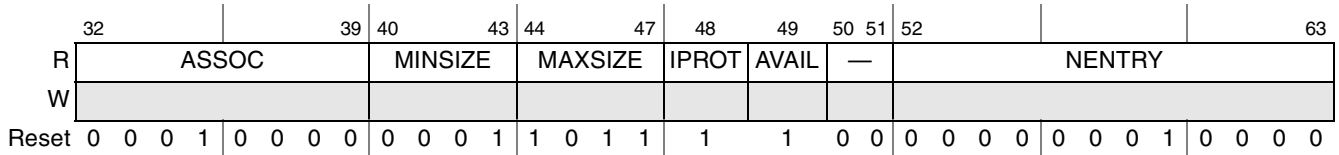
**Table 6-26. TLB0CFG Field Descriptions (continued)**

Bits	Name	Description
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB0 0x100 LB0 contains 256 entries 0x200 TLB0 contains 512 entries (e500v2 only)

### 6.12.4.2 TLB1 Configuration Register 1 (TLB1CFG)

SPR 689

Access: Supervisor read only



**Figure 6-42. TLB Configuration Register 1 (TLB1CFG)**

**Table 6-27. TLB1CFG Field Descriptions**

Bits	Name	Description
32–39	ASSOC	Associativity of TLB1 0x10 indicates associativity is 16
40–43	MINSIZE	Minimum page size of TLB1 0x1 indicates smallest page size is 4K
44–47	MAXSIZE	Maximum page size of TLB1 0x9 Indicates maximum page size is 256 Mbyte 0xB Indicates maximum page size is 4 Gbytes (e500v2 only)
48	IPROT	Invalidate protect capability of TLB1 1 Indicates that TLB1 supports invalidate protection capability
49	AVAIL	Page size availability of TLB1 1 Indicates all page sizes between MINSIZE and MAXSIZE supported
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB1 0x010: TLB1 contains 16 entries

## 6.12.5 MMU Assist Registers

### 6.12.5.1 MAS Register 0 (MAS0)

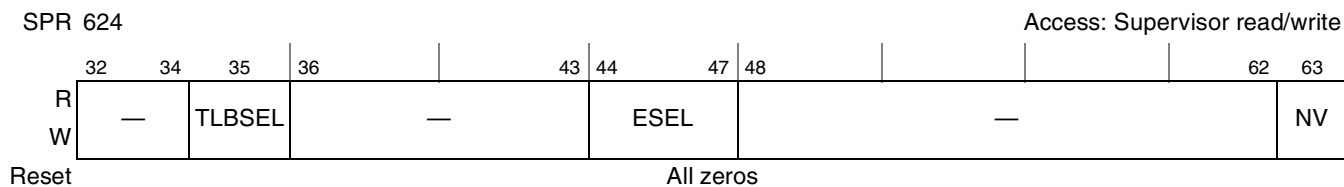


Figure 6-43. MAS Register 0 (MAS0)

Table 6-28. MAS0 Field Descriptions—MMU Read/Write and Replacement Control

Bits	Name	Descriptions
32–34	—	Reserved, should be cleared.
35	TLBSEL	Selects TLB for access 0 TLB0 1 TLB1
36–43	—	Reserved, should be cleared.
44–47	ESEL	Entry select. Number of entry in selected array to be used for <b>tlbwe</b> . This field is also updated on TLB error exceptions (misses), and <b>tlbsx</b> hit and miss cases. Only certain bits are valid, depending on the array selected in TLBSEL. Other bits should be 0. For the e500, ESEL serves as the way select for the corresponding TLB as follows: When TLBSEL = 00 (TLB0 selected), bits 46–47 are used (and bits 44–45 should be cleared). This field selects between way 0, 1, 2, or 3 of TLB0. EA bits 45–51 from MAS2[EPN] are used to index into the TLB to further select the entry for the operation. Note that for the e500v1, bit 47 selects either way 0 or way 1, and bit 46 should remain cleared. When TLBSEL = 01 (TLB1 selected), all four bits are used to select one of 16 entries in the array.
48–62	—	Reserved, should be cleared.
63	NV	Next victim. Next victim bit value to be written to TLB0[NV] on execution of <b>tlbwe</b> . This field is also updated on TLB error exceptions (misses), <b>tlbsx</b> hit and miss cases and on execution of <b>tlbre</b> . This field is updated based on the calculated next victim bit for TLB0 (based on the round-robin replacement algorithm.) Note that this field is not defined for operations that specify TLB1 (when TLBSEL = 01).

### 6.12.5.2 MAS Register 1 (MAS1)

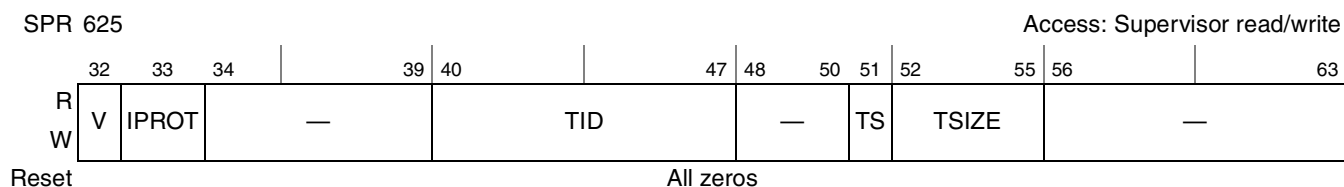


Figure 6-44. MAS Register 1 (MAS1)

Table 6-29. MAS1 Field Descriptions—Descriptor Context and Configuration Control

Bits	Name	Descriptions												
32	V	TLB valid bit 0 This TLB entry is invalid. 1 This TLB entry is valid.												
33	IPROT	Invalidate protect. Set to protect this TLB entry from invalidate operations due the execution of <b>tlbiva[x]</b> (TLB1 only). Note that not all TLB arrays are necessarily protected from invalidation with IPROT. Arrays that support invalidate protection are denoted as such in the TLB configuration registers. 0 Entry is not protected from invalidation 1 Entry is protected from invalidation.												
34–39	—	Reserved, should be cleared.												
40–47	TID	Translation identity. An 8-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the three virtual address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.												
48–50	—	Reserved, should be cleared.												
51	TS	Translation space. This bit is compared with the IS or DS fields of the MSR (depending on the type of access) to determine if this TLB entry may be used for translation.												
52–55	TSIZE	Translation size. Defines the TLB entry page size. For arrays that contain fixed-size TLB entries, TSIZE is ignored. For variable page size arrays, the page size is 4 <sup>TSIZE</sup> Kbytes. The e500 supports the following sizes:  <table style="width: 100%; border: none;"> <tr> <td style="width: 50%;">0001 4 Kbytes</td> <td style="width: 50%;">0111 16 Mbytes</td> </tr> <tr> <td>0010 16 Kbytes</td> <td>1000 64 Mbytes</td> </tr> <tr> <td>0011 64 Kbytes</td> <td>1001 256 Mbytes</td> </tr> <tr> <td>0100 256 Kbytes</td> <td>1010 1 Gbyte (e500v2 only)</td> </tr> <tr> <td>0101 1 Mbyte</td> <td>1011 4 Gbytes (e500v2 only)</td> </tr> <tr> <td>0110 4 Mbytes</td> <td></td> </tr> </table>	0001 4 Kbytes	0111 16 Mbytes	0010 16 Kbytes	1000 64 Mbytes	0011 64 Kbytes	1001 256 Mbytes	0100 256 Kbytes	1010 1 Gbyte (e500v2 only)	0101 1 Mbyte	1011 4 Gbytes (e500v2 only)	0110 4 Mbytes	
0001 4 Kbytes	0111 16 Mbytes													
0010 16 Kbytes	1000 64 Mbytes													
0011 64 Kbytes	1001 256 Mbytes													
0100 256 Kbytes	1010 1 Gbyte (e500v2 only)													
0101 1 Mbyte	1011 4 Gbytes (e500v2 only)													
0110 4 Mbytes														
56–63	—	Reserved, should be cleared.												

### 6.12.5.3 MAS Register 2 (MAS2)

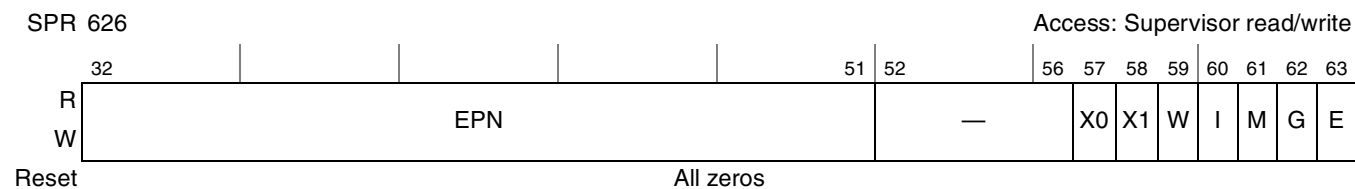


Figure 6-45. MAS Register 2 (MAS2)

Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes

Bits	Name	Description
32–51	EPN	Effective page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–56	—	Reserved for implementation-specific use
57	X0	Implementation-dependent page attribute
58	X1	Implementation-dependent page attribute



Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes (continued)

Bits	Name	Description
59	W	Write-through 0 This page is considered write-back with respect to the caches in the system. 1 All stores performed to this page are written through the caches to main memory.
60	I	Caching-inhibited 0 Accesses to this page are considered cacheable. 1 The page is considered caching-inhibited. All loads and stores to the page bypass the caches and are performed directly to main memory.
61	M	Memory coherency required 0 Memory coherency is not required. 1 Memory coherency is required. This allows loads and stores to this page to be coherent with loads and stores from other processors (and devices) in the system, assuming all such devices are participating in the coherency protocol.
62	G	Guarded 0 Accesses to this page are not guarded and can be performed before it is known if they are required by the sequential execution model. 1 All loads and stores to this page that miss in the L1 cache are performed without speculation (that is, they are known to be required). Speculative loads can be performed if they hit in the L1 cache. In addition, accesses to caching-inhibited pages are performed using only the memory element that is explicitly specified.
63	E	Endianness. Determines endianness for the corresponding page. Little-endian operation is true little endian, which differs from the modified little-endian byte-ordering model optionally available in previous devices that implement the original PowerPC architecture. See the <i>PowerPC™ e500 Core Family Reference Manual</i> for more information. 0 The page is accessed in big-endian byte order. 1 The page is accessed in true little-endian byte order.

### 6.12.5.4 MAS Register 3 (MAS3)

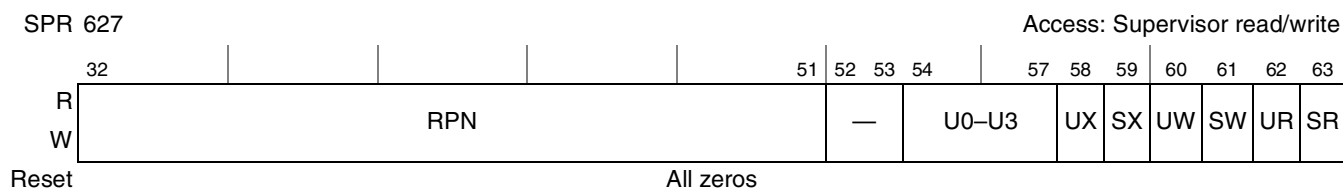
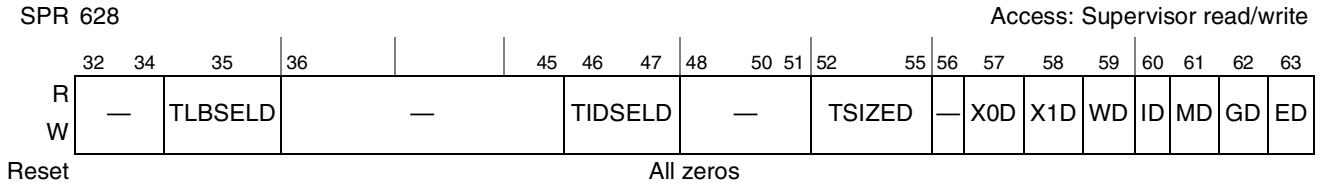


Figure 6-46. MAS Register 3 (MAS3)

Table 6-31. MAS3 Field Descriptions—RPN and Access Control

Bits	Name	Description
32–51	RPN	Real page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared. For the e500v2, the 4 high-order bits of RPN are stored in MAS7[RPN].
52–53	—	Reserved, should be cleared.
54–57	U0–U3	User attribute bits. Associated with a TLB entry and can be used by system software. For example, they can hold information useful to a page-scanning algorithm or mark more abstract page attributes.
58–63	PERMIS	Permission bits (UX, SX, UW, SW, UR, SR). User and supervisor read, write, and execute permission bits.

### 6.12.5.5 MAS Register 4 (MAS4)

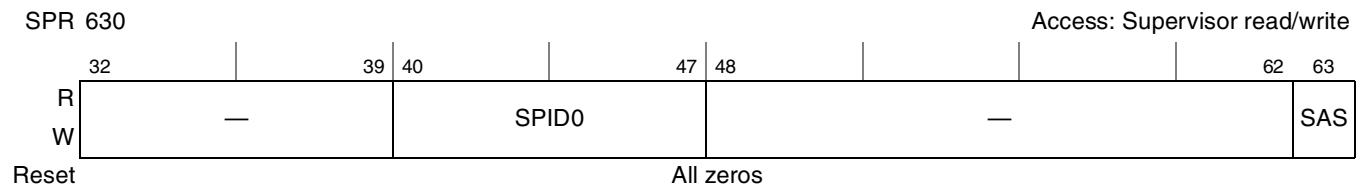


**Figure 6-47. MAS Register 4 (MAS4)**

**Table 6-32. MAS4 Field Descriptions—Hardware Replacement Assist Configuration**

Bits	Name	Description
32–34	—	Reserved, should be cleared.
35	TLBSELD	TLBSEL default value. The default value to be loaded in MAS0[TLBSEL] on a TLB miss exception. 0 TLB0 1 TLB1
36–45	—	Reserved, should be cleared.
46–47	TIDSELD	TID default selection value. A 2-bit field that specifies which of the current PID registers should be used to load the MAS1[TID] field on a TLB miss exception.  The e500 implementation defines this field as follows: 00 PID0 01 PID1 10 PID2 11 TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
52–55	TSIZED	Default TSIZE value. Specifies the default value to be loaded into MAS1[TSIZE] on a TLB miss exception.
56	—	Reserved, should be cleared.
57	X0D	Default X0 value. Specifies the default value to be loaded into MAS2[X0] on a TLB miss exception.
58	X1D	Default X1 value. Specifies the default value to be loaded into MAS2[X1] on a TLB miss exception.
59	WD	Default W value. Specifies the default value to be loaded into MAS2[W] on a TLB miss exception.
60	ID	Default I value. Specifies the default value to be loaded into MAS2[I] on a TLB miss exception.
61	MD	Default M value. Specifies the default value to be loaded into MAS2[M] on a TLB miss exception.
62	GD	Default G value. Specifies the default value to be loaded into MAS2[G] on a TLB miss exception.
63	ED	Default E value. Specifies the default value to be loaded into MAS2[E] on a TLB miss exception.

### 6.12.5.6 MAS Register 6 (MAS6)



**Figure 6-48. MAS Register 6 (MAS6)**

Table 6-33. MAS6—TLB Search Context Register 0

Bits	Name	Comments, or Function when Set
32–39	—	Reserved, should be cleared.
40–47	SPID0	Specifies the PID value (recent value of PID0) used when searching the TLB during execution of <b>tlbsx</b> .
48–62	—	Reserved, should be cleared.
63	SAS	Address space (AS) value for searches. Specifies the value of AS used when searching the TLB (during execution of <b>tlbsx</b> ).

### 6.12.5.7 MAS Register 7 (MAS7)

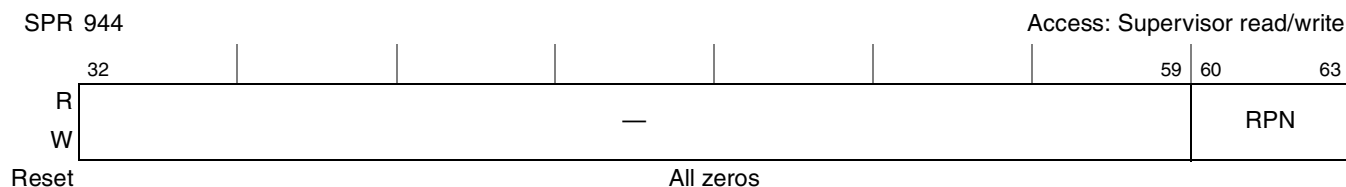


Figure 6-49. MAS Register 7 (MAS7)

Table 6-34. MAS 7 Field Descriptions—High Order RPN

Bits	Name	Description
32–63	RPN	Real page number. Four high-order bits of the RPN. The 20 low-order bits of RPN are in MAS3.

## 6.13 Debug Registers

### 6.13.1 Debug Control Registers (DBCR0–DBCR2)

#### 6.13.1.1 Debug Control Register 0 (DBCR0)

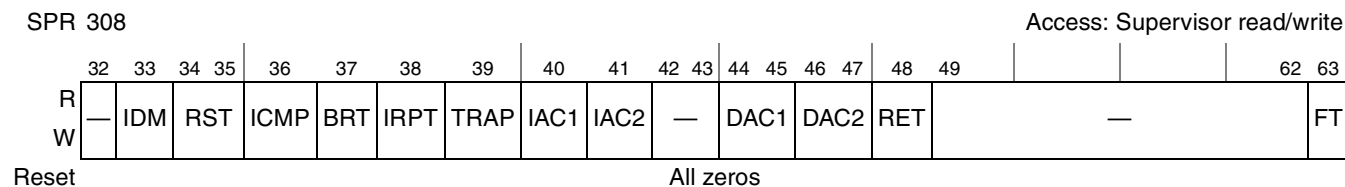


Figure 6-50. Debug Control Register 0 (DBCR0)

Table 6-35. DBCR0 Field Descriptions

Bits	Name	Description
32	—	Reserved, should be cleared.
33	IDM	Internal debug mode 0 Debug interrupts are disabled. No debug interrupts are taken and debug events are not logged. 1 If MSR[DE] = 1, the occurrence of a debug event or the recording of an earlier debug event in the DBSR when MSR[DE] = 0 or DBCR0[IDM] = 0 causes a debug interrupt. <b>Programming note:</b> Software must clear debug event status in the DBSR in the debug interrupt handler when a debug interrupt is taken before re-enabling interrupts through MSR[DE]. Otherwise, redundant debug interrupts are taken for the same debug event.
34–35	RST	Reset. The e500 implements these bits as follows: 0x Default (No action) 1x Causes a hard reset if MSR[DE] and DBCR0[IDM] are set. Always cleared on subsequent cycle. This causes a hard reset to the core only.
36	ICMP	Instruction completion debug event enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled Note: Instruction completion does not cause an ICMP debug event if MSR[DE] = 0.
37	BRT	Branch taken debug event enable 0 BRT debug events are disabled 1 BRT debug events are enabled Note: Taken branches do not cause a BRT debug event if MSR[DE] = 0.
38	IRPT	Interrupt taken debug event enable. This bit affects only noncritical interrupts. 0 IRPT debug events are disabled 1 IRPT debug events are enabled
39	TRAP	Trap debug event enable 0 TRAP debug events cannot occur 1 TRAP debug events can occur
40	IAC1	Instruction address compare 1 debug event enable 0 IAC1 debug events cannot occur 1 IAC1 debug events can occur
41	IAC2	Instruction address compare 2 debug event enable 0 IAC2 debug events cannot occur 1 IAC2 debug events can occur
42–43	—	Reserved, should be cleared.
44–45	DAC1	Data address compare 1 debug event enable 00 DAC1 debug events cannot occur 01 DAC1 debug events can occur only if a store-type data storage access 10 DAC1 debug events can occur only if a load-type data storage access 11 DAC1 debug events can occur on any data storage access
46–47	DAC2	Data address compare 2 debug event enable 00 DAC2 debug events cannot occur 01 DAC2 debug events can occur only if a store-type data storage access 10 DAC2 debug events can occur only if a load-type data storage access 11 DAC2 debug events can occur on any data storage access

Table 6-35. DBCR0 Field Descriptions (continued)

Bits	Name	Description
48	RET	Return debug event enable 0 RET debug events cannot occur 1 RET debug events can occur Note: An <b>rfdi</b> does not cause an RET debug event if MSR[DE] = 0 at the time that <b>rfdi</b> executes.
49–62	—	Reserved, should be cleared.
63	FT	Freeze timers on debug event 0 Enable clocking of timers 1 Disable clocking of timers if any DBSR bit is set (except MRR)

### 6.13.1.2 Debug Control Register 1 (DBCR1)

SPR 309

Access: Supervisor read/write

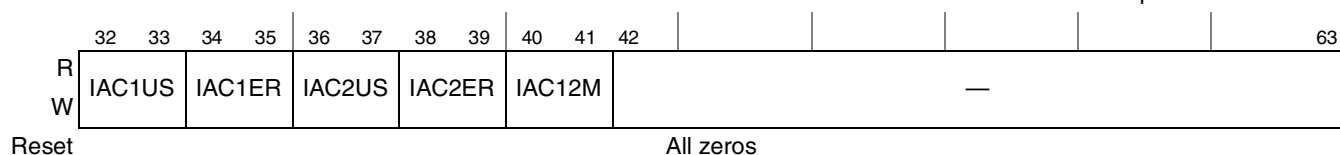


Figure 6-51. Debug Control Register 1 (DBCR1)

Table 6-36. DBCR1 Field Descriptions

Bits	Name	Description
32–33	IAC1US	Instruction address compare 1 user/supervisor mode 00 IAC1 debug events can occur 01 Reserved 10 IAC1 debug events can occur only if MSR[PR] = 0 11 IAC1 debug events can occur only if MSR[PR] = 1
34–35	IAC1ER	Instruction address compare 1 effective/real mode 00 IAC1 debug events are based on effective addresses 01 Reserved on the e500 10 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 1
36–37	IAC2US	Instruction address compare 2 user/supervisor mode 00 IAC2 debug events can occur 01 Reserved 10 IAC2 debug events can occur only if MSR[PR] = 0 11 IAC2 debug events can occur only if MSR[PR] = 1
38–39	IAC2ER	Instruction address compare 2 effective/real mode 00 IAC2 debug events are based on effective addresses 01 Reserved on the e500 10 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 1

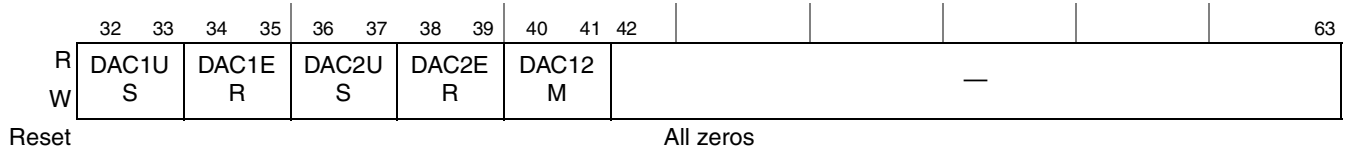
**Table 6-36. DBCR1 Field Descriptions (continued)**

Bits	Name	Description
40–41	IAC12M	<p>Instruction address compare 1/2 mode</p> <p>00 Exact address compare. IAC1 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC2.</p> <p>01 Address bit match. IAC1 and IAC2 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, plus ANDed with the contents of IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined.</p> <p>10 Inclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined.</p> <p>11 Exclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined.</p>
42–63	—	Reserved, should be cleared.

### 6.13.1.3 Debug Control Register 2 (DBCR2)

SPR 310

Access: Supervisor read/write

**Figure 6-52. Debug Control Register 2 (DBCR2)****Table 6-37. DBCR2 Field Descriptions**

Bits	Name	Description
32–33	DAC1US	<p>Data address compare 1 user/supervisor mode</p> <p>00 DAC1 debug events can occur</p> <p>01 Reserved</p> <p>10 DAC1 debug events can occur only if MSR[PR] = 0.</p> <p>11 DAC1 debug events can occur only if MSR[PR] = 1.</p>
34–35	DAC1ER	<p>Data address compare 1 effective/real mode</p> <p>00 DAC1 debug events are based on effective addresses.</p> <p>01 Reserved on the e500</p> <p>10 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 0.</p> <p>11 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 1.</p>
36–37	DAC2US	<p>Data address compare 2 user/supervisor mode</p> <p>00 DAC2 debug events can occur.</p> <p>01 Reserved</p> <p>10 DAC2 debug events can occur only if MSR[PR] = 0.</p> <p>11 DAC2 debug events can occur only if MSR[PR] = 1.</p>

Table 6-37. DBCR2 Field Descriptions (continued)

Bits	Name	Description
38–39	DAC2ER	Data address compare 2 effective/real mode 00 DAC2 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
40–41	DAC12M	Data address compare 1/2 mode 00 Exact address compare. DAC1 debug events can occur only if the address of the data storage access is equal to the value specified in DAC1. DAC2 debug events can occur only if the address of the data storage access is equal to the value specified in DAC2. 01 Address bit match. DAC1 and DAC2 debug events can occur only if the address of the data storage access, ANDed with the contents of DAC2 are equal to the contents of DAC1, also ANDed with the contents of DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 10 Inclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 11 Exclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

### 6.13.2 Debug Status Register (DBSR)

SPR: 304

Access: Supervisor w1c

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R	IDE	UDE	MRR		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—		DAC1 R	DAC1W	DAC2 R	DAC2W
W	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c
Reset	0	0	undefined		0	0	0	0	0	0	0	0	0	0	0	0
	48	49														63
R	RET	—														
W	w1c															
Reset	All zeros															

Figure 6-53. Debug Status Register (DBSR)

Table 6-38. DBSR Field Descriptions

Bits	Name	Description
32	IDE	Imprecise debug event. Set if MSR[DE] = 0 and a debug event causes its respective DBSR bit to be set. Functions as write-one-to-clear.
33	UDE	Unconditional debug event. Set if an unconditional debug event occurred. Functions as write-one-to-clear. If UDE (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: <u>MSR[DE]DBCR0[IDM]Action</u> X 0 No action. 0 1 UDE is set. 1 1 UDE is set and a debug interrupt is taken.
34–35	MRR	Most recent reset. Functions as write-one-to-clear. Undefined at power-on. The e500 implements HRESET as follows: 0x No hard reset occurred since this bit was last cleared by software. 1x The previous reset was a hard reset.
36	ICMP	Instruction complete debug event. Set if an instruction completion debug event occurred and DBCR0[ICMP] = 1. Functions as write-one-to-clear.
37	BRT	Branch taken debug event. Set if a branch taken debug event occurred (DBCR0[BRT] = 1). Functions as write-one-to-clear.
38	IRPT	Interrupt taken debug event. Set if an interrupt taken debug event occurred (DBCR0[IRPT] = 1). Functions as write-one-to-clear.
39	TRAP	Trap instruction debug event. Set if a trap instruction debug event occurred (DBCR0[TRAP] = 1). Functions as write-one-to-clear.
40	IAC1	Instruction address compare 1 debug event. Set if an IAC1 debug event occurred (DBCR0[IAC1] = 1). Functions as write-one-to-clear.
41	IAC2	Instruction address compare 2 debug event. Set if an IAC2 debug event occurred (DBCR0[IAC2] = 1). Functions as write-one-to-clear.
42–43	—	Reserved, should be cleared
44	DAC1R	Data address compare 1 read debug event. Set if a read-type DAC1 debug event occurred (DBCR0[DAC1] = 10 or 11). Functions as write-one-to-clear.
45	DAC1W	Data address compare 1 write debug event. Set if a write-type DAC1 debug event occurred (DBCR0[DAC1] = 01 or 11). Functions as write-one-to-clear.
46	DAC2R	Data address compare 2 read debug event. Set if a read-type DAC2 debug event occurred (DBCR0[DAC2] = 10 or 11). Functions as write-one-to-clear.
47	DAC2W	Data address compare 2 write debug event. Set if a write-type DAC2 debug event occurred (DBCR0[DAC2] = 01 or 11). Functions as write-one-to-clear.
48	RET	Return debug event. Set if a return debug event occurred (DBCR0[RET] = 1). Functions as write-one-to-clear.
49–63	—	Reserved, should be cleared.



### 6.13.3 Instruction Address Compare Registers (IAC1–IAC2)

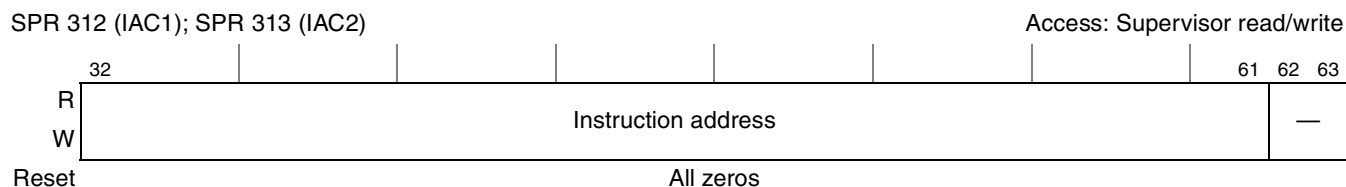


Figure 6-54. Instruction Address Compare Registers (IAC1–IAC2)

### 6.13.4 Data Address Compare Registers (DAC1–DAC2)

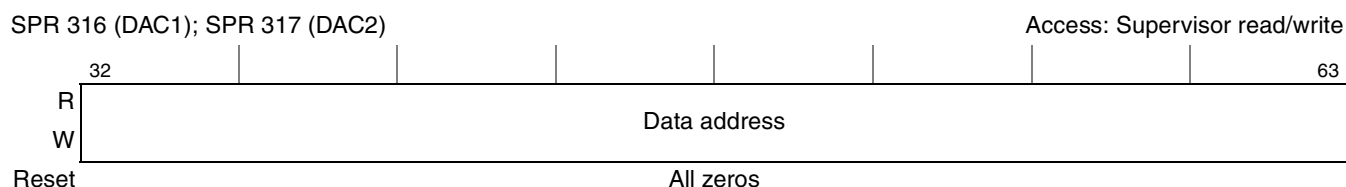


Figure 6-55. Data Address Compare Registers (DAC1–DAC2)

## 6.14 Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

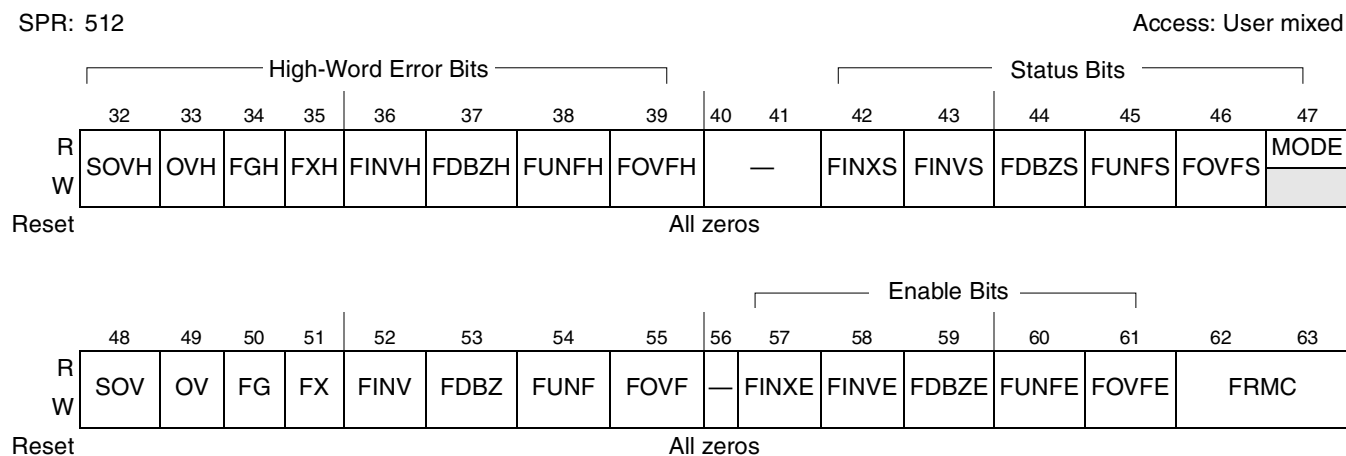


Figure 6-56. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Table 6-39. SPEFSCR Field Descriptions

Bits	Name	Function
32	SOVH	Summary integer overflow high. Set whenever an instruction (except <b>mtspr</b> ) sets OVH. SOVH remains set until it is cleared by an <b>mtspr[SPEFSCR]</b> .
33	OVH	Integer overflow high. An overflow occurred in the upper half of the register while executing a SPE integer instruction

Table 6-39. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
34	FGH	Embedded floating-point guard bit high. Floating-point guard bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
35	FXH	Embedded floating-point sticky bit high. Floating bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
36	FINVH	Embedded floating-point invalid operation error high. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
37	FDBZH	Embedded floating-point divide by zero error high. Set if the dividend is non-zero and the divisor is zero.
38	FUNFH	Embedded floating-point underflow error high
39	FOVFH	Embedded floating-point overflow error high
40–41	—	Reserved, should be cleared.
42	FINXS	Embedded floating-point inexact sticky. $FINXS = FINXS \mid FGH \mid FXH \mid FG \mid FX$
43	FINVS	Embedded floating-point invalid operation sticky. Location for software to use when implementing true IEEE floating point.
44	FDBZS	Embedded floating-point divide by zero sticky. $FDBZS = FDBZS \mid FDBZH \mid FDBZ$
45	FUNFS	Embedded floating-point underflow sticky. Storage location for software to use when implementing true IEEE floating point.
46	FOVFS	Embedded floating-point overflow sticky. Storage location for software to use when implementing true IEEE floating point.
47	MODE	Embedded floating-point mode (read only on e500)
48	SOV	Integer summary overflow. Set whenever an SPE instruction (except <b>mtspr</b> ) sets OV. SOV remains set until it is cleared by <b>mtspr[SPEFSCR]</b> .
49	OV	Integer overflow. An overflow occurred in the lower half of the register while a SPE integer instruction is being executed.
50	FG	Embedded floating-point guard bit. Floating-point guard bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
51	FX	Embedded floating-point sticky bit. Floating bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
52	FINV	Embedded floating-point invalid operation error. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
53	FDBZ	Embedded floating-point divide by zero error. Set of the dividend is non-zero and the divisor is zero.
54	FUNF	Embedded floating-point underflow error
55	FOVF	Embedded floating-point overflow error
56	—	Reserved, should be cleared.
57	FINXE	Embedded floating-point inexact enable

Table 6-39. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
58	FINVE	Embedded floating-point invalid operation/input error exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FINV or FINVH is set by a floating-point instruction.
59	FDBZE	Embedded floating-point divide-by-zero exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FDBZ or FDBZH is set by a floating-point instruction.
60	FUNFE	Embedded floating-point underflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FUNF or FUNFH is set by a floating-point instruction.
61	FOVFE	Embedded floating-point overflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FOVF or FOVFH is set by a floating-point instruction.
62–63	FRMC	Embedded floating-point rounding mode control 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward –infinity

### 6.14.1 Accumulator (ACC)

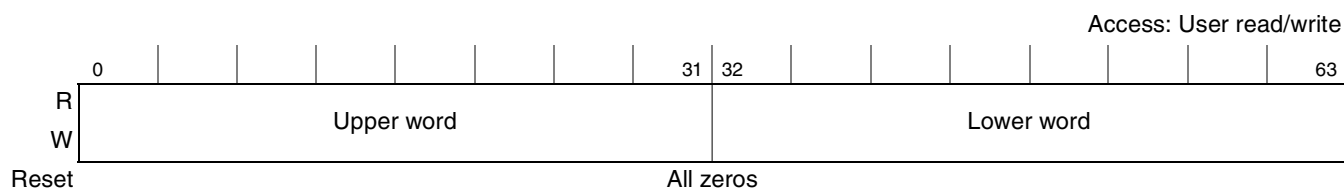


Figure 6-57. Accumulator (ACC)

Table 6-40. ACC Field Descriptions

Bits	Name	Function
0–31	Upper word	Holds the upper-word accumulate value for SPE multiply with accumulate instructions
32–63	Lower word	Holds the lower-word accumulate value for SPE multiply with accumulate instructions

## 6.15 Performance Monitor Registers (PMRs)

**Table 6-41. Supervisor-Level PMRs (PMR[5] = 1)**

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
PMC0	Performance monitor counter 0	16	00000	10000	6.15.4/6-52
PMC1	Performance monitor counter 1	17	00000	10001	
PMC2	Performance monitor counter 2	18	00000	10010	
PMC3	Performance monitor counter 3	19	00000	10011	
PMGC0	Performance monitor global control register 0	400	01100	10000	6.15.1/6-49
PMLCa0	Performance monitor local control a0	144	00100	10000	6.15.2/6-50
PMLCa1	Performance monitor local control a1	145	00100	10001	
PMLCa2	Performance monitor local control a2	146	00100	10010	
PMLCa3	Performance monitor local control a3	147	00100	10011	
PMLCb0	Performance monitor local control b0	272	01000	10000	6.15.3/6-51
PMLCb1	Performance monitor local control b1	273	01000	10001	
PMLCb2	Performance monitor local control b2	274	01000	10010	
PMLCb3	Performance monitor local control b3	275	01000	10011	

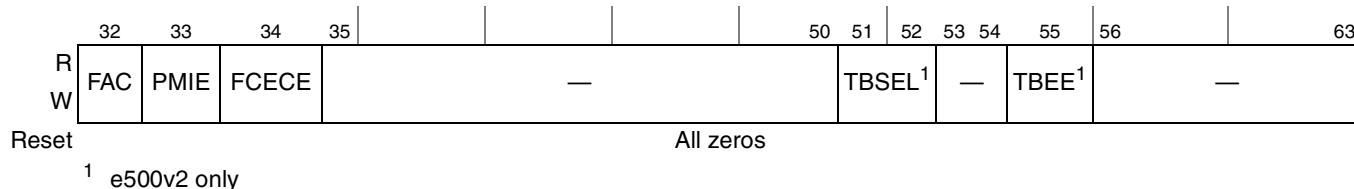
**Table 6-42. User-Level PMRs (PMR[5] = 0) (Read Only)**

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
UPMC0	User performance monitor counter 0	0	00000	00000	6.15.4/6-52
UPMC1	User performance monitor counter 1	1	00000	00001	
UPMC2	User performance monitor counter 2	2	00000	00010	
UPMC3	User performance monitor counter 3	3	00000	00011	
UPMLCa0	User performance monitor local control a0	128	00100	00000	6.15.3/6-51
UPMLCa1	User performance monitor local control a1	129	00100	00001	
UPMLCa2	User performance monitor local control a2	130	00100	00010	
UPMLCa3	User performance monitor local control a3	131	00100	00011	
UPMLCb0	User performance monitor local control b0	256	01000	00000	6.15.3/6-51
UPMLCb1	User performance monitor local control b1	257	01000	00001	
UPMLCb2	User performance monitor local control b2	258	01000	00010	
UPMLCb3	User performance monitor local control b3	259	01000	00011	
UPMGC0	User performance monitor global control register 0	384	01100	00000	6.15.2/6-50

## 6.15.1 Global Control Register 0 (PMGC0, UPMGC0)

PMGC0 (PMR400)  
UPMGC0 (PMR384)

Access: PMGC0: Supervisor- read/write  
UPMGC0: Supervisor/user read only



**Figure 6-58. Performance Monitor Global Control Register 0 (PMGC0),  
User Performance Monitor Global Control Register 0 (UPMGC0)**

**Table 6-43. PMGC0 Field Descriptions**

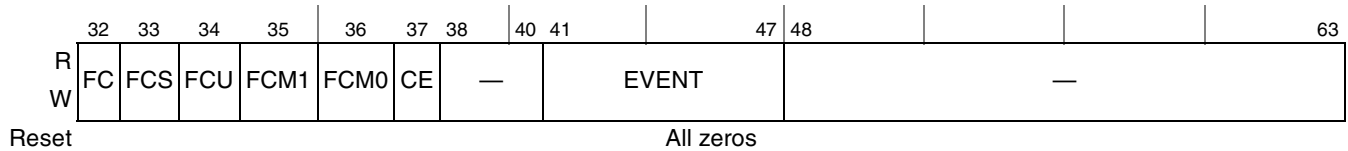
Bits	Name	Description
32	FAC	Freeze all counters. When FAC is set by hardware or software, PMLCx[FC] maintains its current value until it is changed by software. 0 The PMCs are incremented (if permitted by other PM control bits). 1 The PMCs are not incremented.
33	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs.
34	FCECE	Freeze counters on enabled condition or event 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0[FAC] is set. It is up to software to clear FAC.
35–50	—	Reserved, should be cleared.
51–52	TBSEL	Time base selector. Selects the time base bit that can cause a time base transition event (the event occurs when the selected bit changes from 0 to 1). 00 TB[63] (TBL[31]) 01 TB[55] (TBL[23]) 10 TB[51] (TBL[19]) 11 TB[47] (TBL[15]) Time base transition events can be used to periodically collect information about processor activity. In multiprocessor systems in which TB registers are synchronized among processors, time base transition events can be used to correlate the performance monitor data obtained by the several processors. For this use, software must specify the same TBSEL value for all processors in the system. Because the time-base frequency is implementation-dependent, software should invoke a system service program to obtain the frequency before choosing a value for TBSEL.
53–54	—	Reserved, should be cleared.

**Table 6-43. PMGC0 Field Descriptions (continued)**

Bits	Name	Description
55	TBEE	Time base transition event exception enable. 0 Exceptions from time base transition events are disabled. 1 Exceptions from time base transition events are enabled. A timebase transition is signaled to the performance monitor if the TB bit specified in PMGC0[TBSEL] changes from 0 to 1. Timebase transition events can be used to freeze the counters (PMGC0[FCECE]) or signal an exception (PMGC0[PMIE]). Changing PMGC0[TBSEL] while PMGC0[TBEE] is enabled may cause a false 0 to 1 transition that signals the specified action (freeze, exception) to occur immediately. Although the interrupt signal condition may occur with MSR[EE] = 0, the interrupt cannot be taken until MSR[EE] = 1.
56–63	—	Reserved, should be cleared.

### 6.15.2 Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3)

PMLCa0 (PMR144)                      UPMLCa0 (PMR128)                      Access: PMLCa0–PMLCa3: Supervisor read/write  
 PMLCa1 (PMR145)                      UPMLCa1 (PMR129)                      UPMLCa0–UPMLCa3: Supervisor/user read only  
 PMLCa2 (PMR146)                      UPMLCa2 (PMR130)  
 PMLCa3 (PMR147)                      UPMLCa3 (PMR131)



**Figure 6-59. Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)**

**Table 6-44. PMLCa0–PMLCa3 Field Descriptions**

Bits	Name	Description
32	FC	Freeze counter 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented.
33	FCS	Freeze counter in supervisor state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 0.
34	FCU	Freeze counter in user state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 1.
35	FCM1	Freeze counter while mark = 1 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 1.
36	FCM0	Freeze counter while mark = 0 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 0.

Table 6-44. PMLCa0–PMLCa3 Field Descriptions (continued)

Bits	Name	Description
37	CE	Condition enable 0 PMCx overflow conditions cannot occur (PMCx cannot cause interrupts, cannot freeze counters) 1 Overflow conditions occur when the most-significant bit of PMCx is equal to 1. It is recommended that CE be cleared when counter PMCx is selected for chaining.
38–40	—	Reserved, should be cleared.
41–47	EVENT	Event selector. Up to 128 events selectable. These events are described in the <i>PowerPC™ e500 Core Family Reference Manual</i> .
48–63	—	Reserved, should be cleared.

### 6.15.3 Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3)

PMLCb0 (PMR272) UPMLCb0 (PMR256)  
PMLCb1 (PMR273) UPMLCb1 (PMR257)  
PMLCb2 (PMR274) UPMLCb2 (PMR258)  
PMLCb3 (PMR275) UPMLCb3 (PMR259)

Access: PMLCb0–PMLCb3: Supervisor read/write  
UPMLCb0–UPMLCb3: Supervisor/user read only



Figure 6-60. Local Control B Registers (PMLCb0–PMLCb3)/User Local Control B Registers (UPMLCb0–UPMLCb3)

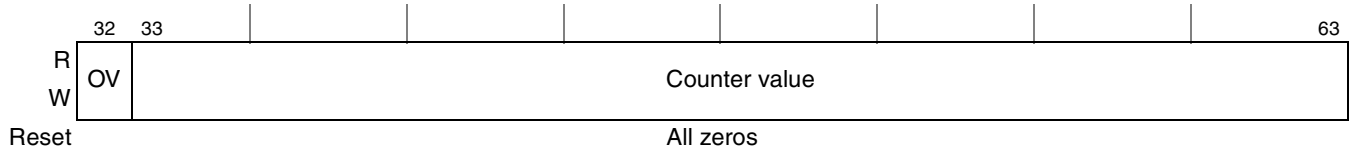
Table 6-45. PMLCb0–PMLCb3 Field Descriptions

Bits	Name	Description
32–52	—	Reserved, should be cleared.
53–55	THRESHMUL	Threshold multiple 000 Threshold field is multiplied by 1 ( $PMLCb_n[THRESHOLD] \times 1$ ) 001 Threshold field is multiplied by 2 ( $PMLCb_n[THRESHOLD] \times 2$ ) 010 Threshold field is multiplied by 4 ( $PMLCb_n[THRESHOLD] \times 4$ ) 011 Threshold field is multiplied by 8 ( $PMLCb_n[THRESHOLD] \times 8$ ) 100 Threshold field is multiplied by 16 ( $PMLCb_n[THRESHOLD] \times 16$ ) 101 Threshold field is multiplied by 32 ( $PMLCb_n[THRESHOLD] \times 32$ ) 110 Threshold field is multiplied by 64 ( $PMLCb_n[THRESHOLD] \times 64$ ) 111 Threshold field is multiplied by 128 ( $PMLCb_n[THRESHOLD] \times 128$ )
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the threshold value are counted. Such events are implementation-dependent as are the dimension (for example duration in cycles) and granularity with which the value is interpreted. By varying the value, software can obtain a profile of the event characteristics subject to thresholding. For example, if PMC1 is configured to count cache misses that last longer than the threshold value, software can obtain the distribution of cache miss durations for a given program by monitoring the program repeatedly using a different threshold each time.

### 6.15.4 Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3)

PMC0 (PMR16) UPMC0 (PMR0)  
 PMC1 (PMR17) UPMC1 (PMR1)  
 PMC2 (PMR18) UPMC2 (PMR2)  
 PMC3 (PMR19) UPMC3 (PMR3)

Access: PMC0–PMC3: Supervisor read/write  
 UPMC0–UPMC3: Supervisor/user read only



**Figure 6-61. Performance Monitor Counter Registers (PMC0–PMC3)/User Performance Monitor Counter Registers (UPMC0–UPMC3)**

**Table 6-46. PMC0–PMC3 Field Descriptions**

Bits	Name	Description
32	OV	Overflow. When this bit is set, it indicates this counter reaches its maximum value.
33–63	Counter value	Indicates the number of occurrences of the specified event



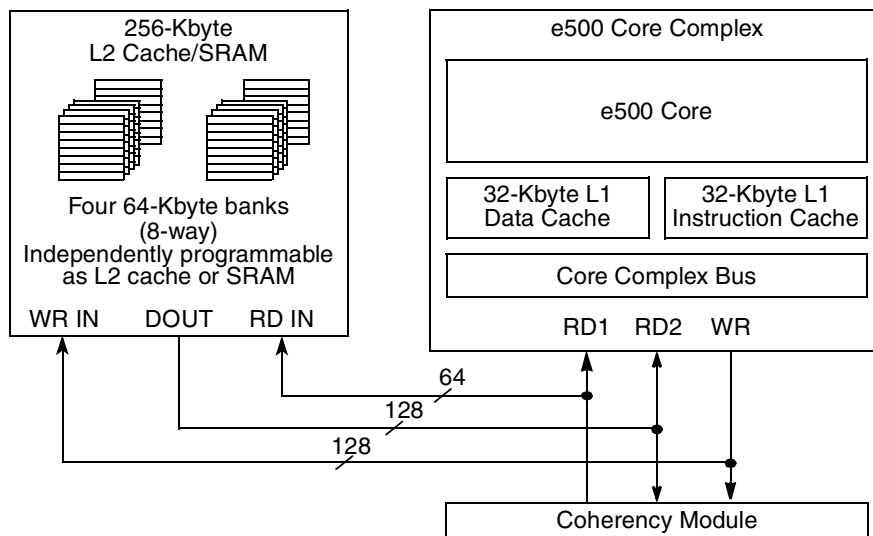
# Chapter 7

## L2 Look-Aside Cache/SRAM

This chapter describes the organization of the on-chip L2/SRAM, cache coherency rules, cache line replacement algorithm, cache control instructions, and various cache operations. It also describes the interaction between the L2/SRAM and the e500 core complex.

### 7.1 L2 Cache Overview

The integrated 256-Kbyte L2 cache is organized as 1024 eight-way sets of 32-byte cache lines based on 36-bit physical addresses, as shown in [Figure 7-1](#).



**Figure 7-1. L2 Cache/SRAM Configuration**

The SRAM can be configured with memory-mapped registers as externally accessible memory-mapped SRAM in addition to or instead of cache. The L2 cache can operate in the following modes, described in [Section 7.2, “L2 Cache and SRAM Organization”](#):

- Full cache mode (256-Kbyte cache).
- Full memory-mapped SRAM mode (256-Kbyte SRAM mapped as a single 256-Kbyte block or two 128-Kbyte blocks)
- Partial SRAM and partial cache mode, in which one eighth, one quarter, or one half the total on-chip memory can be allocated to 1 or 2 SRAM regions.

## 7.1.1 L2 Cache and SRAM Features

The L2 cache has the following characteristics:

- Supports 36-bit address space
- Write-through, front-side cache
  - Front-side design provides easier cache access for the I/O masters, such as Ethernet
  - Write-through design is more efficient on the processor bus for front-side caches
- Valid, locked, and stale states (no modified state)
- Two input data buses (64 and 128 bits wide) and one output data bus (128 bits wide)
- All accesses are fully pipelined and non-blocking (allows hits under misses)
- 256-Kbyte array organized as 1024 eight-way sets of 32-byte cache lines
- Eight-way set associativity with a pseudo-LRU (7-bit) replacement algorithm.
  - High level of associativity yields good performance even with many lines locked or used as SRAM regions
- I/O devices can store data into the cache in a process called ‘stashing.’
  - Stashing is indicated for global I/O writes either by a transaction attribute or by a programmable memory range
  - Regions of the cache can be reserved exclusively for stashing to prevent pollution of processor cache regions.
- Processor L2 cache regions are configurable to allocate instructions, data, or both.
  - External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing).
    - 1, 2, or 4 ways can be configured for stashing only
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag arrays use 21 tag bits and 1 tag parity bit per line.
- Multiple cache locking methods supported
  - Individual line locks are set and cleared by using e500 cache locking APU instructions—Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**).
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers ( $L2CEWAR_n$ ).
  - The entire cache can be locked by setting a configuration register appropriately.
- Lock clearing methods
  - Individual locks can be cleared by cache-locking APU instructions—Data Cache Block Lock Clear (**dcblc**) and Instruction Cache Block Lock Clear (**icblc**)—or by a snooped flush unless entire cache is locked.
  - Flash clearing of all instruction and/or data locks is done by writes to configuration registers.
  - An unlock attribute can be attached to a read instruction.

- Error injection modes supported for testing error handling

SRAM features include the following:

- SRAM regions are created by configuring 1, 2, 4 or 8 ways of each set to be reserved for memory-mapped SRAM.
- Regions can reside at any location in the memory map aligned to the SRAM size.
- SRAM memory is byte addressable; for accesses of less than a cache line, ECC is updated using read-modify-write transactions.
- I/O devices access SRAM regions by marking transactions as snoopable (global).

Table 1 lists the possible L2 cache/SRAM configurations.

**Table 1. Available L2 Cache/SRAM Configurations**

Cache	Stash-only Region	SRAM Region 1	SRAM Region 2
256 Kbytes	—	—	—
224 Kbytes	—	32 Kbytes	—
	32 Kbytes	—	—
192 Kbytes	—	64 Kbytes	—
	—	32 Kbytes	32 Kbytes
	32 Kbytes	32 Kbytes	—
	64 Kbytes	—	—
160 Kbytes	32 Kbytes	64 Kbytes	—
	—	32 Kbytes	32 Kbytes
	64 Kbytes	32 Kbytes	—
128 Kbytes	—	128 Kbytes	—
	—	64 Kbytes	64 Kbytes
	64 Kbytes	64 Kbytes	—
	—	32 Kbytes	32 Kbytes
	128 Kbytes	—	—
96 Kbytes	32 Kbytes	128 Kbytes	—
	—	64 Kbytes	64 Kbytes
	128 Kbytes	32 Kbytes	—
64 Kbytes	64 Kbytes	128 Kbytes	—
	—	64 Kbytes	64 Kbytes
	128 Kbytes	64 Kbytes	—
	—	32 Kbytes	32 Kbytes
—	—	256 Kbytes	—
	—	128 Kbytes	128 Kbytes
	128 Kbytes	128 Kbytes	—
	—	64 Kbytes	64 Kbytes

## 7.2 L2 Cache and SRAM Organization

The on-chip memory array has four banks, each containing 256 sets of eight cache blocks (or ‘ways’), as shown in Figure 7-2. Each block consists of 32 bytes of data and a tag.

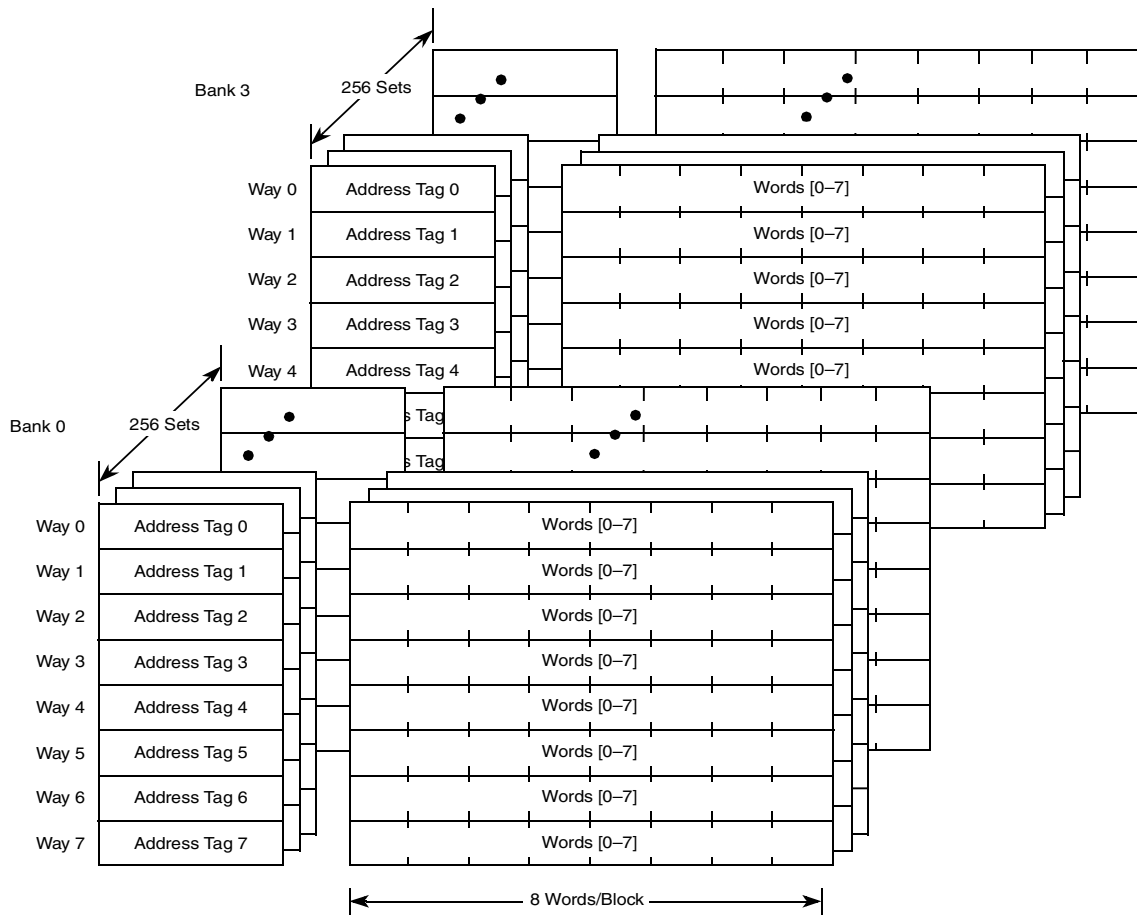
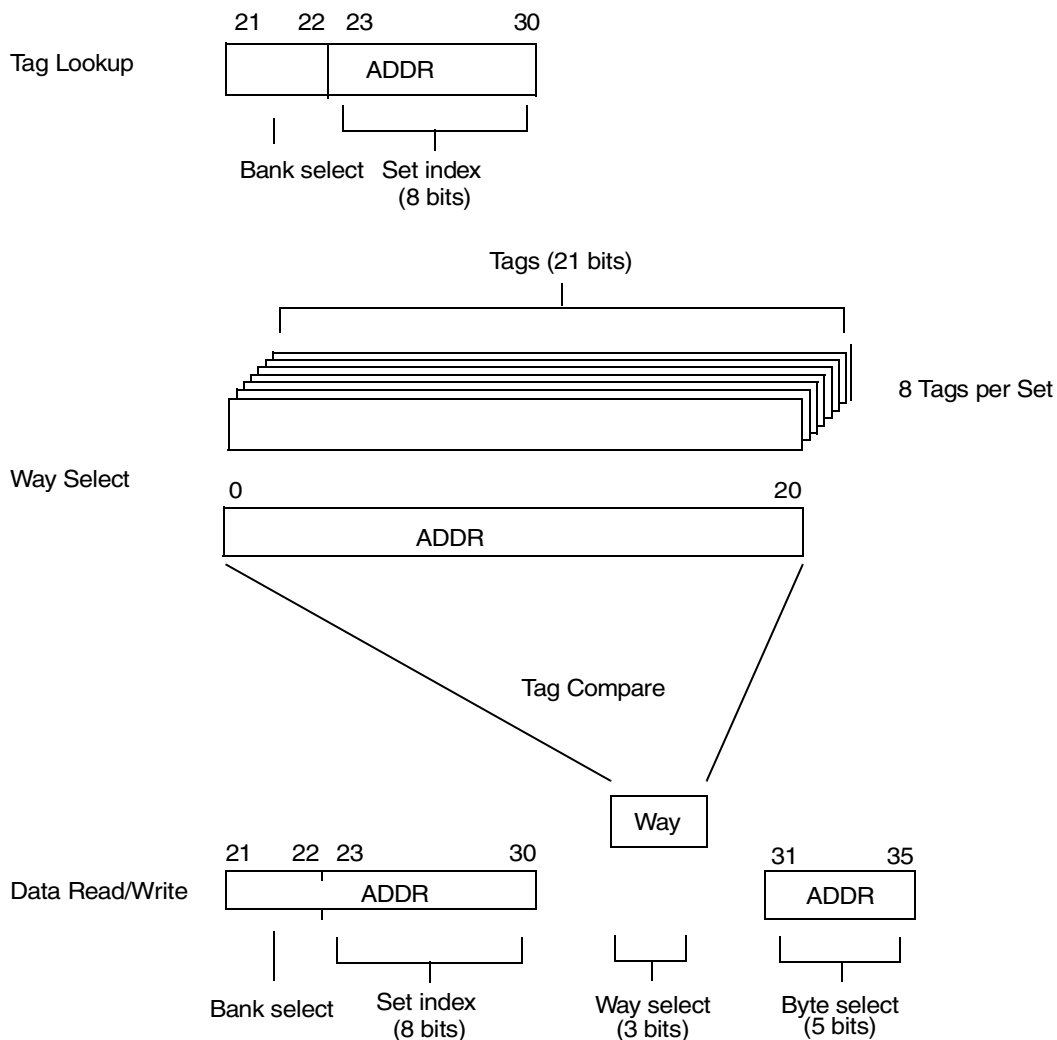


Figure 7-2. Cache Organization

## 7.2.1 Accessing the On-Chip Array as an L2 Cache

Figure 7-3 shows how physical address bits are used to access the L2 cache.

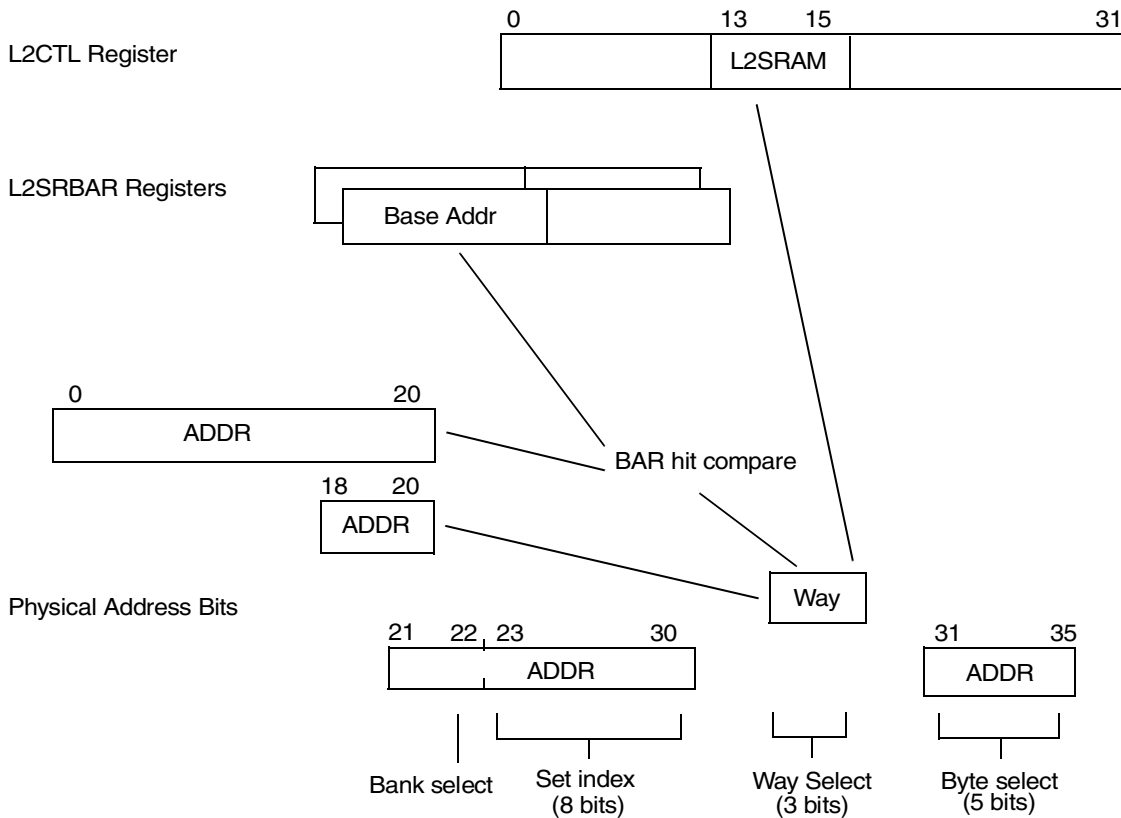


**Figure 7-3. Physical Address Usage for L2 Cache Accesses**

Physical address bits 21–30 identify the bank and set of the tag and data. Physical address bits 0–20 are compared against the tags of all eight ways. A match of a valid tag selects a 32-byte block of data (or way) within the set. Physical address bits 31–35 identify the byte or bytes of data within the block.

## 7.2.2 Accessing the On-Chip Array as an SRAM

When all or part of the array is dedicated to memory mapped SRAM, individual ways of each set are reserved for that purpose. SRAM accesses use physical address bits 18–20 in conjunction with the SRAM mode to select a way of the indexed set.



**Figure 7-4. Physical Address Usage for SRAM Accesses**

The mapping of address bits and SRAM mode to a way select is shown below in [Table 7-2](#). SRAM size is reflected in L2CTL[L2SIZ].

**Table 7-2. Way Selection for SRAM Accesses**

Description	L2SRAM	BAR 0 Hit	BAR 1 Hit	Addr[18–20]	Way Select
No SRAM	000	—	—	—	—
Entire Array is SRAM (single 256-KB SRAM if L2SIZ=256 KB)	001	1	0	000	0
				001	1
				010	2
				011	3
				100	4
				101	5
				110	6
				111	7

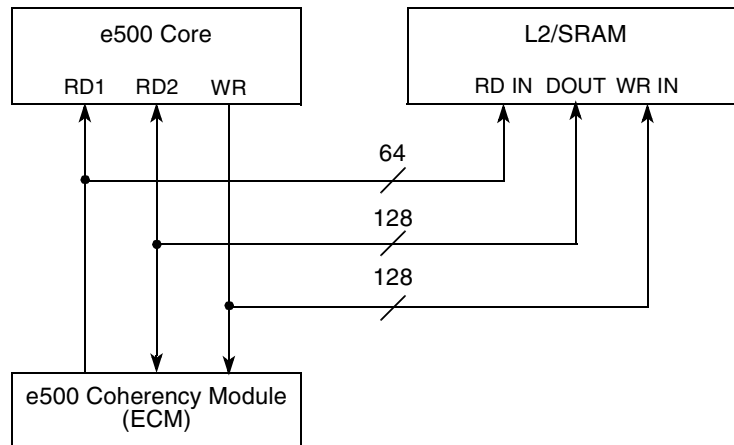
Table 7-2. Way Selection for SRAM Accesses (continued)

Description	L2SRAM	BAR 0 Hit	BAR 1 Hit	Addr[18–20]	Way Select
One half of array is an SRAM (single 128-KB SRAM if L2SIZ=256 KB)	010	1	0	x00	0
				x01	1
				x10	2
				x11	3
Both halves of array are SRAM (two 128-KB SRAM if L2SIZ=256 KB)	011	1	0	x00	0
				x01	1
				x10	2
				x11	3
	0	1	x00	4	
			x01	5	
			x10	6	
			x11	7	
One quarter of the array is SRAM (single 64-KB SRAM if L2SIZ=256 KB)	100	1	0	xx0	0
				xx1	1
Two quarters of the array are SRAMs (single 64-KB SRAM if L2SIZ=256 KB)	101	1	0	xx0	0
				xx1	1
	0	1	xx0	2	
			xx1	3	
One eighth of the array is an SRAM (single 32-KB SRAM if L2SIZ=256 KB)	110	1	0	—	0
Two eighths of the array are SRAM (single 32-KB SRAM if L2SIZ=256 KB)	111	1	0	—	0
		0	1	—	1

### 7.2.3 Connection of the On-Chip Memory to the System

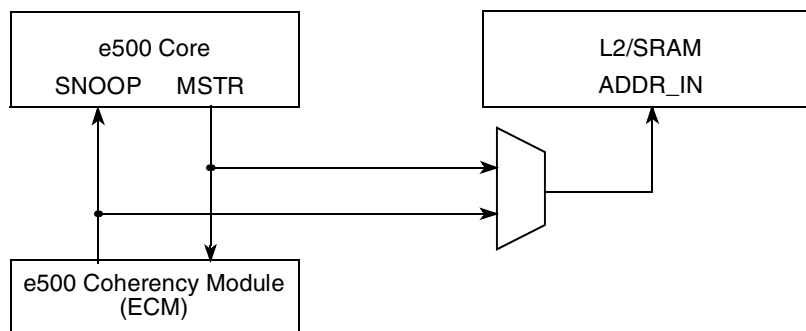
The e500 core connects to the L2 cache and the system interface through the high-speed core complex bus (CCB). The e500 core and the L2 cache connect to the rest of the integrated device through the e500 coherency module (ECM). [Figure 7-5](#) shows the data connections of the e500 core and L2/SRAM. The e500 core can simultaneously read 128 bits of data from the L2/SRAM, read 64 bits of data from the system interface, and write 128 bits of data to the L2/SRAM and/or system interface.

The L2/SRAM can be accessed by the e500 core or the system interface through the ECM. The L2 cache does not initiate transactions. [Figure 7-5](#) shows the data bus connections of the e500 core and L2/SRAM.



**Figure 7-5. Data Bus Connection of CCB**

[Figure 7-6](#) shows address connections of the e500 core and L2/SRAM.



**Figure 7-6. Address Bus Connection of CCB**

In SRAM mode, if a non-cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non-cache-line write transaction cannot be allocated in the L2.

## 7.3 Memory Map/Register Definition

[Table 7-3](#) shows the memory map for the L2/SRAM registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.



Table 7-3. L2/SRAM Memory-Mapped Registers

Offset	Register	Access	Reset	Section/Page
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-10
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.2.1/7-13
0x2_0014	L2CEWAREA0—L2 cache external write address register extended address 0	R/W	0x0000_0000	7.3.1.2.2/7-14
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.2.3/7-14
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.2.1/7-13
0x2_0024	L2CEWAREA1—L2 cache external write address register extended address 1	R/W	0x0000_0000	7.3.1.2.2/7-14
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.2.3/7-14
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.2.1/7-13
0x2_0034	L2CEWAREA2—L2 cache external write address register extended address 2	R/W	0x0000_0000	7.3.1.2.2/7-14
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.2.3/7-14
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.2.1/7-13
0x2_0044	L2CEWAREA3—L2 cache external write address register extended address 3	R/W	0x0000_0000	7.3.1.2.2/7-14
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.2.3/7-14
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.3.1/7-16
0x2_0104	L2SRBAREA0—L2 memory-mapped SRAM base address register extended address 0	R/W	0x0000_0000	7.3.1.3.2/7-17
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.3.1/7-16
0x2_010C	L2SRBAREA1—L2 memory-mapped SRAM base address register extended address 1	R/W	0x0000_0000	7.3.1.3.2/7-17
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.4.1/7-18
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.4.1/7-18
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.4.1/7-18
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.4.2/7-20
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.4.2/7-20
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.4.2/7-20
0x2_0E40	L2ERRDET—L2 error detect register	w1c	0x0000_0000	7.3.1.4.2/7-20
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.4.2/7-20
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.4.2/7-20
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.4.2/7-20
0x2_0E50	L2ERRADDRH—L2 error address capture register high	R	0x0000_0000	7.3.1.4.2/7-20
0x2_0E54	L2ERRADDRL—L2 error address capture register low	R	0x0000_0000	7.3.1.4.2/7-20
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.4.2/7-20

### 7.3.1 L2/SRAM Register Descriptions

The following sections describe registers that control and configure the L2/SRAM array.

#### 7.3.1.1 L2 Control Register (L2CTL)

The L2 control register (L2CTL), shown in Figure 7-7, controls configuration and operation of the L2/SRAM array. The sequence for modifying L2CTL is as follows:

1. **mbar**
2. **isync**
3. **stw** (WIMG = 01xx) CCSRBAR+0x2\_0000
4. **lwz** (WIMG = 01xx) CCSRBAR+0x2\_0000
5. **mbar**

Offset 0x2\_0000

Access: Read/Write

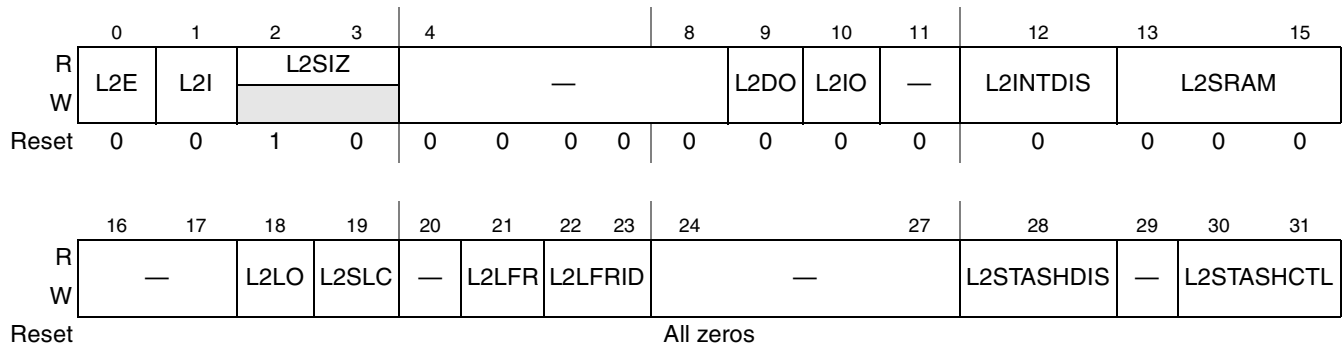


Figure 7-7. L2 Control Register (L2CTL)

Table 7-4 describes L2CTL fields.

Table 7-4. L2CTL Field Descriptions

Bits	Name	Description
0	L2E	L2 enable. Used to enable the L2 array (cache or memory-mapped SRAM). 0 The L2 SRAM (cache and memory-mapped SRAM) is disabled and is not accessed for reads, snoops, or writes. Setting the L2 flash invalidate bit (L2I) is allowed. 1 The L2 SRAM (cache or memory-mapped SRAM) is enabled. Note that L2I can be set regardless of the value of L2E.
1	L2I	L2 flash invalidate. 0 The L2 status and LRU bits are not being cleared. 1 Setting L2I invalidates the L2 cache globally by clearing the all the L2 status bits, as well as the LRU algorithm. Memory-mapped SRAM is unaffected. Data to memory-mapped SRAM are unaffected by the flash invalidate. The hardware automatically clears L2I when the invalidate is complete.
2–3	L2SIZ	L2 SRAM size (read only). Indicates the total available size of on-chip memory array (to be configured as cache or memory-mapped SRAM). 00 Reserved 01 256 Kbyte 10 Reserved 11 Reserved

Table 7-4. L2CTL Field Descriptions (continued)

Bits	Name	Description
4–8	—	Reserved
9	L2DO	<p>L2 data-only. Reserved in full memory-mapped SRAM mode. L2DO may be changed while the L2 is enabled or disabled.</p> <p>0 The L2 cache allocates entries for instruction fetches that miss in the L2.  1 The L2 cache allocates entries for processor data loads that miss in the L2 and for processor L1 castouts but does not allocate entries for instruction fetches that miss in the L2. Instruction accesses that hit in the L2, data accesses, and accesses from the system (including I/O stash writes) are unaffected.</p> <p>Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.</p>
10	L2IO	<p>L2 instruction-only. Reserved in full memory-mapped SRAM mode. Causes the L2 cache to allocate lines for instruction cache transactions only. L2IO may be changed while the L2 is enabled or disabled.</p> <p>0 The L2 cache entries are allocated for data loads that miss in the L2 and for processor L1 castouts.  1 The L2 cache allocates entries for instruction fetch misses, but does not allocate entries for processor data transactions. Data accesses that hit in the L2, instruction accesses, and accesses from the system (including I/O stash writes) are unaffected.</p> <p>Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.</p>
11	—	Reserved
12	L2INTDIS	<p>Cache read intervention disable. Reserved for full memory-mapped SRAM mode. Used to disable cache read intervention. May be changed while the L2 is enabled or disabled.</p> <p>0 Cache intervention is enabled. The ECM ensures that if a data read from another device hits in the L2 cache, it is serviced from the L2 cache.  1 Cache intervention is disabled</p>
13–15	L2SRAM	<p>L2 SRAM configuration. Determines the L2 cache/memory-mapped SRAM allocation of the on-chip memory array. SRAM size depends on the value of L2SIZ. Since L2SIZ is 256 Kbytes, SRAM can have sizes from 32 Kbytes to 256 Kbytes.</p> <p>000 No SRAM. Entire array is cache.  001 Entire array is a single SRAM (256-Kbyte SRAM for L2SIZ = 256 Kbytes)  010 One half of the array is an SRAM (128-Kbyte SRAM for L2SIZ = 256 Kbytes)  011 Both halves of the array are SRAMs (two 128-Kbyte SRAMs for L2SIZ = 256 Kbytes)  100 One quarter of the array is an SRAM (one 64-Kbyte SRAM for L2SIZ = 256 Kbytes)  101 Two quarters of the array are SRAMs (two 64-Kbyte SRAMs for L2SIZ = 256 Kbytes)  110 One eighth of the array is an SRAM (one 32-Kbyte SRAM for L2SIZ = 256 Kbytes)  111 Two eighths of the array are SRAMs (two 32-Kbyte SRAMs for L2SIZ = 256 Kbytes)</p> <p>For one SRAM region L2SRBAR0 is used and for two SRAM regions L2SRBAR0 and L2SRABAR1 are used. Regions of the array that are not allocated to SRAMs will be used as cache memory.</p> <p>To change these bits, the L2 must be disabled (L2CTL[L2E] = 0).</p> <p>Note that when setting L2SRAM after cache has been enabled, L2I should be set as well. The fields can be set simultaneously, and this step is not needed if SRAM size is getting smaller.</p>
16–17	—	Reserved

Table 7-4. L2CTL Field Descriptions (continued)

Bits	Name	Description
18	L2LO	L2 cache lock overflow. Reserved in full memory-mapped SRAM mode. This sticky bit is set if an overlock condition is detected in the L2 cache. A lock overflow is triggered either by executing instruction or data cache block touch and lock set instructions or by performing L2 cache external writes with lock set. If all ways are locked and an attempt to stash is made, the stash is not allocated. 0 The L2 cache did not encounter a lock overflow. L2LO is cleared only by software. 1 The L2 cache encountered a lock overflow condition.
19	L2SLC	L2 snoop lock clear. This sticky bit is set if a snoop invalidated a locked data cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. L2SLC is reserved in full memory-mapped SRAM mode. 0 A snoop did not invalidate a locked L2 cache line. L2SLC is cleared only by software. 1 The L2 cache encountered a snoop that invalidated a locked line.
20	—	Reserved
21	L2LFR	L2 cache lock bits flash reset. The L2 cache must be enabled (L2CTL[L2E] = 1) for reset to occur. This field is reserved in full memory-mapped SRAM mode. 0 The L2 cache lock bits are not cleared or the clear operation completed. 1 A reset operation is issued that clears each L2 cache line's lock bits. Depending on the L2LFRID value, data or instruction locks, or both, can be reset. Cache access is blocked during this time. After L2LFR is set, the L2 cache unit automatically clears L2LFR when the reset operation is complete (if L2CTL[L2E] is set).
22–23	L2LFRID	L2 cache lock bits flash reset select instruction or data. Indicates whether data or instruction lock bits or both are reset. 00 Not used 01 Reset data locks if L2LFR = 1. 10 Reset instruction locks if L2LFR = 1. 11 Reset both data and instruction locks if L2LFR = 1.
24–27	—	Reserved
28	L2STASHDIS	L2 stash allocate disable. Disables allocation of lines for stashing. 0 The L2 cache will allocate lines for global writes that hit in a stash range or that have the stashing attribute set. 1 The L2 will not allocate lines for stashed writes. <b>Note:</b> This bit does NOT affect the updating of lines that are already resident in the cache and have the stash attribute set or hit a stash range. Such lines will be updated even if this bit is set. To change this bit, the L2 must be disabled (L2CTL[L2E] = 0).

Table 7-4. L2CTL Field Descriptions (continued)

Bits	Name	Description
29	—	Reserved
30–31	L2STASHCTL	<p>L2 stash configuration. This field reserves regions of the cache for stash-only operation. That is, blocks of each cache set are reserved so that they can only be allocated for stash data. If such a region is created, processor reads and writes will not be allocated into this region; it can only be populated by stash writes. Similarly, stash writes will only be allocated into this region. This prevents processor and stashed I/O data from polluting one another.</p> <p>00 No stash-only region. Stashed writes will be allocated across the entire cache and can evict processor data and can be evicted by processor data.</p> <p>01 One half of the array is a stash-only cache (way4, way5, way6 &amp; way7 of each set)</p> <p>10 One quarter of the array is a stash-only cache (way6 &amp; way7 of each set)</p> <p>11 One eighth of the array is a stash-only cache (way7 of each set)</p> <p>Like L2SRAM configuration, stash-only regions subtract from the amount of the on-chip memory that is available to the processor as cache. If the L2SRAM configuration uses the entire on-chip memory array as SRAM, then no stash-only region can be created.</p> <p>To change these bits, the L2 must be disabled (L2CTL[L2E] = 0). This field has no effect if the L2STASHDIS bit is set.</p>

### 7.3.1.2 L2 Cache External Write Registers

The device supports allocating and locking L2 cache lines from external agents such as PCI. This functionality is called stashing. Four sets of registers are provided to support this feature; each set has three registers that specify a programmed memory range that can be locked with a snoop write transaction. All three registers in a set must be configured in order to use an external write address.

These registers are the L2 cache external write address registers 0–3 (L2CEWAR<sub>n</sub>), the L2 cache external write address registers extended address 0–3 (L2CEWAREA<sub>n</sub>), and the L2 cache external write control registers 0–3 (L2CEWCR<sub>n</sub>). L2CEWAR<sub>n</sub> contain the lower 24 bits of the external write base address and L2CEWAREA<sub>n</sub> contain the upper 4 bits. The base address specified in the address registers must be naturally aligned to the window size in the corresponding control register.

Further details on the locations and fields of these registers are given in the following sections.

#### 7.3.1.2.1 L2 Cache External Write Address Registers 0–3 (L2CEWAR<sub>n</sub>)

The L2CEWAR<sub>n</sub> registers contain the lower 24 bits of the 28-bit L2 cache external write base address. Each of these registers has identical fields, as shown in [Figure 7-8](#).

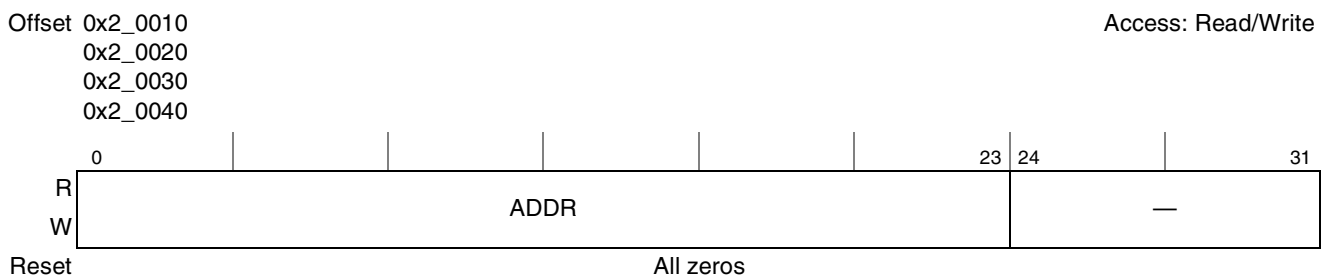


Figure 7-8. Cache External Write Address Registers (L2CEWAR<sub>n</sub>)

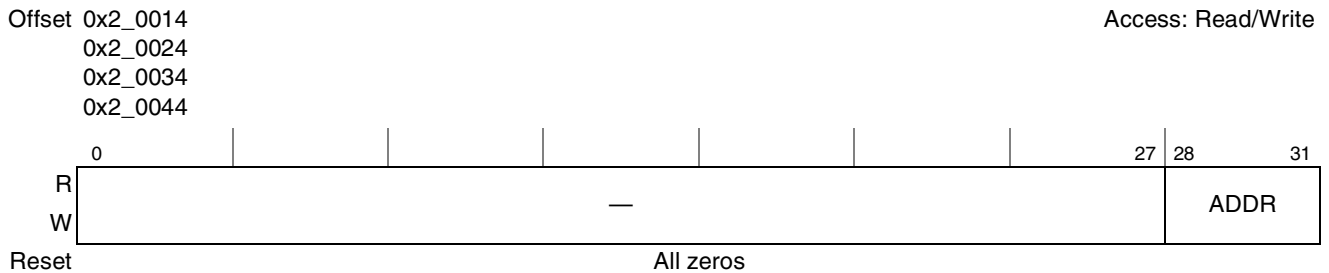
Table 7-5 describes L2CEWAR<sub>n</sub> fields.

**Table 7-5. L2CEWAR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–23	ADDR	Contains the lower 24 bits of the 28-bit L2 cache external write base address. Note that the upper 4 bits of the base address are in L2CEWAREA <sub>n</sub> [ADDR].
24–31	—	Reserved

### 7.3.1.2.2 L2 Cache External Write Address Registers Extended Address 0–3 (L2CEWAREA<sub>n</sub>)

The L2 cache external write address registers extended address (L2CEWAREA<sub>n</sub>), shown in Figure 7-9, contain the upper 4 bits of the 28-bit L2 cache external write base address.



**Figure 7-9. Cache External Write Address Registers Extended Address (L2CEWAREA<sub>n</sub>)**

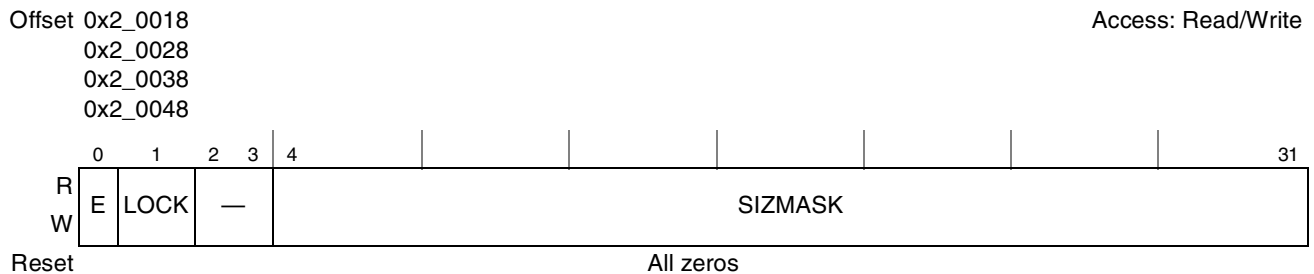
Table 7-6 describes the fields of L2CEWAREA<sub>n</sub>.

**Table 7-6. L2CEWAREA<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Contains the upper 4 bits of the L2 cache external write base address. Note that the rest of the base address is in L2CEWAR <sub>n</sub> [ADDR].

### 7.3.1.2.3 L2 Cache External Write Control Registers 0–3 (L2CEWCR<sub>n</sub>)

The L2CEWAR<sub>n</sub>/L2CEWAREA<sub>n</sub> address registers work with the L2 cache external write control registers 0–3 (L2CEWCR<sub>n</sub>), shown in Figure 7-10, to control cache external write functionality.



**Figure 7-10. Cache External Write Control Registers (L2CEWCR<sub>0</sub>–L2CEWCR<sub>3</sub>)**

The L2CEWCR $n$  registers contain identical fields, which are described in [Table 7-7](#).

**Table 7-7. L2CEWCR $n$  Field Descriptions**

Bits	Name	Description																														
0	E	External write enable. An external write matching the address window defined by L2CEWAR $n$ /L2CEWAREA $n$ /L2CEWCR $n$ is allocated or updated in the L2 cache. 0 External writes for the L2CEWAR $n$ /L2CEWAREA $n$ /L2CEWCR $n$ set are disabled. 1 External writes are enabled for the L2CEWAR $n$ /L2CEWAREA $n$ /L2CEWCR $n$ set.																														
1	LOCK	Lock lines in the targeted cache. An external write matching the address window defined by L2CEWAR $n$ /L2CEWAREA $n$ /L2CEWCR $n$ is locked in the L2 cache when it is allocated or updated. 0 The locked bit is not set when a line is allocated unless explicitly specified by transaction attributes. 1 Cache lines are allocated as locked. A hit to a valid, unlocked line sets the lock.																														
2–3	—	Reserved																														
4–31	SIZMASK	Mask size. Defines the size of the naturally aligned address region for cache external writes. The address region must be aligned to a boundary that is a multiple of the mask size. Any value not listed below is illegal and produces boundedly undefined results. <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 150px;">1111 1111 1111 1111 1111 1111 1111 256 bytes</td> <td>1111 1111 1111 1000 0000 0000 0000 8 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1111 1110 512 bytes</td> <td>1111 1111 1111 0000 0000 0000 0000 16 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1111 1100 1 Kbyte</td> <td>1111 1111 1110 0000 0000 0000 0000 32 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1111 1000 2 Kbytes</td> <td>1111 1111 1100 0000 0000 0000 0000 64 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1111 0000 4 Kbytes</td> <td>1111 1111 1000 0000 0000 0000 0000 128 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1110 0000 8 Kbytes</td> <td>1111 1111 0000 0000 0000 0000 0000 256 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1100 0000 16 Kbytes</td> <td>1111 1110 0000 0000 0000 0000 0000 512 Mbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1111 1000 0000 32 Kbytes</td> <td>1111 1100 0000 0000 0000 0000 0000 1 Gbyte</td> </tr> <tr> <td>1111 1111 1111 1111 1111 0000 0000 64 Kbytes</td> <td>1111 1000 0000 0000 0000 0000 0000 2 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1110 0000 0000 128 Kbytes</td> <td>1111 0000 0000 0000 0000 0000 0000 4 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1100 0000 0000 256 Kbytes</td> <td>1110 0000 0000 0000 0000 0000 0000 8 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1111 1000 0000 0000 512 Kbytes</td> <td>1100 0000 0000 0000 0000 0000 0000 16 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1111 0000 0000 0000 1 Mbyte</td> <td>1000 0000 0000 0000 0000 0000 0000 32 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1110 0000 0000 0000 2 Mbytes</td> <td>0000 0000 0000 0000 0000 0000 0000 64 Gbytes</td> </tr> <tr> <td>1111 1111 1111 1100 0000 0000 0000 4 Mbytes</td> <td></td> </tr> </table>	1111 1111 1111 1111 1111 1111 1111 256 bytes	1111 1111 1111 1000 0000 0000 0000 8 Mbytes	1111 1111 1111 1111 1111 1111 1110 512 bytes	1111 1111 1111 0000 0000 0000 0000 16 Mbytes	1111 1111 1111 1111 1111 1111 1100 1 Kbyte	1111 1111 1110 0000 0000 0000 0000 32 Mbytes	1111 1111 1111 1111 1111 1111 1000 2 Kbytes	1111 1111 1100 0000 0000 0000 0000 64 Mbytes	1111 1111 1111 1111 1111 1111 0000 4 Kbytes	1111 1111 1000 0000 0000 0000 0000 128 Mbytes	1111 1111 1111 1111 1111 1110 0000 8 Kbytes	1111 1111 0000 0000 0000 0000 0000 256 Mbytes	1111 1111 1111 1111 1111 1100 0000 16 Kbytes	1111 1110 0000 0000 0000 0000 0000 512 Mbytes	1111 1111 1111 1111 1111 1000 0000 32 Kbytes	1111 1100 0000 0000 0000 0000 0000 1 Gbyte	1111 1111 1111 1111 1111 0000 0000 64 Kbytes	1111 1000 0000 0000 0000 0000 0000 2 Gbytes	1111 1111 1111 1111 1110 0000 0000 128 Kbytes	1111 0000 0000 0000 0000 0000 0000 4 Gbytes	1111 1111 1111 1111 1100 0000 0000 256 Kbytes	1110 0000 0000 0000 0000 0000 0000 8 Gbytes	1111 1111 1111 1111 1000 0000 0000 512 Kbytes	1100 0000 0000 0000 0000 0000 0000 16 Gbytes	1111 1111 1111 1111 0000 0000 0000 1 Mbyte	1000 0000 0000 0000 0000 0000 0000 32 Gbytes	1111 1111 1111 1110 0000 0000 0000 2 Mbytes	0000 0000 0000 0000 0000 0000 0000 64 Gbytes	1111 1111 1111 1100 0000 0000 0000 4 Mbytes	
1111 1111 1111 1111 1111 1111 1111 256 bytes	1111 1111 1111 1000 0000 0000 0000 8 Mbytes																															
1111 1111 1111 1111 1111 1111 1110 512 bytes	1111 1111 1111 0000 0000 0000 0000 16 Mbytes																															
1111 1111 1111 1111 1111 1111 1100 1 Kbyte	1111 1111 1110 0000 0000 0000 0000 32 Mbytes																															
1111 1111 1111 1111 1111 1111 1000 2 Kbytes	1111 1111 1100 0000 0000 0000 0000 64 Mbytes																															
1111 1111 1111 1111 1111 1111 0000 4 Kbytes	1111 1111 1000 0000 0000 0000 0000 128 Mbytes																															
1111 1111 1111 1111 1111 1110 0000 8 Kbytes	1111 1111 0000 0000 0000 0000 0000 256 Mbytes																															
1111 1111 1111 1111 1111 1100 0000 16 Kbytes	1111 1110 0000 0000 0000 0000 0000 512 Mbytes																															
1111 1111 1111 1111 1111 1000 0000 32 Kbytes	1111 1100 0000 0000 0000 0000 0000 1 Gbyte																															
1111 1111 1111 1111 1111 0000 0000 64 Kbytes	1111 1000 0000 0000 0000 0000 0000 2 Gbytes																															
1111 1111 1111 1111 1110 0000 0000 128 Kbytes	1111 0000 0000 0000 0000 0000 0000 4 Gbytes																															
1111 1111 1111 1111 1100 0000 0000 256 Kbytes	1110 0000 0000 0000 0000 0000 0000 8 Gbytes																															
1111 1111 1111 1111 1000 0000 0000 512 Kbytes	1100 0000 0000 0000 0000 0000 0000 16 Gbytes																															
1111 1111 1111 1111 0000 0000 0000 1 Mbyte	1000 0000 0000 0000 0000 0000 0000 32 Gbytes																															
1111 1111 1111 1110 0000 0000 0000 2 Mbytes	0000 0000 0000 0000 0000 0000 0000 64 Gbytes																															
1111 1111 1111 1100 0000 0000 0000 4 Mbytes																																

### 7.3.1.3 L2 Memory-Mapped SRAM Registers

The registers described in this section, the L2 memory-mapped SRAM base address registers 0–1 (L2SRBAR $n$ ) and the L2 memory-mapped SRAM base address registers extended address 0–1 (L2SRBAREA $n$ ), control the memory-mapped SRAM mode functionality. Together, these two pairs of registers define memory blocks that can be mapped into the L2 cache.

Specified SRAM base addresses must be aligned to the size of the SRAM region. If L2CTL[L2SRAM] specifies one memory-mapped SRAM block, its base address must be written to the pair L2SRBAR0 and L2SRBAREA0; if it specifies two memory-mapped SRAM blocks, L2SRBAR0 and L2SRBAREA0 are used for the first SRAM block and L2SRBAR1 and L2SRBAREA1 are used for the second block.

### 7.3.1.3.1 L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR $n$ )

The L2 memory-mapped SRAM base address registers (L2SRBAR $n$ ), shown in Figure 7-11, contain the lower 18 bits of the 22-bit SRAM base address.

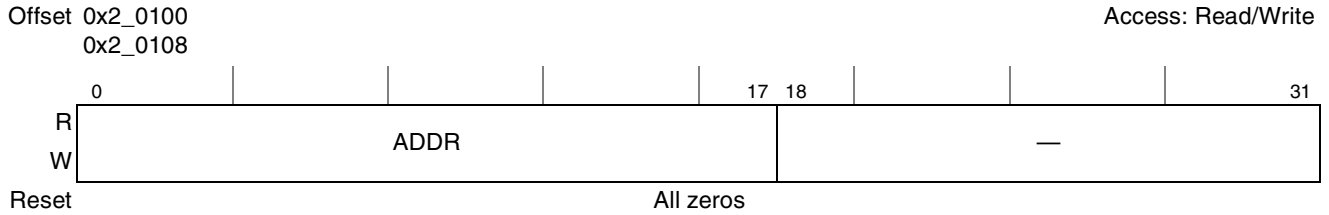


Figure 7-11. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR $n$ )

L2SRBAR bits are described in Table 7-8.

Table 7-8. L2SRBAR $n$  Field Descriptions

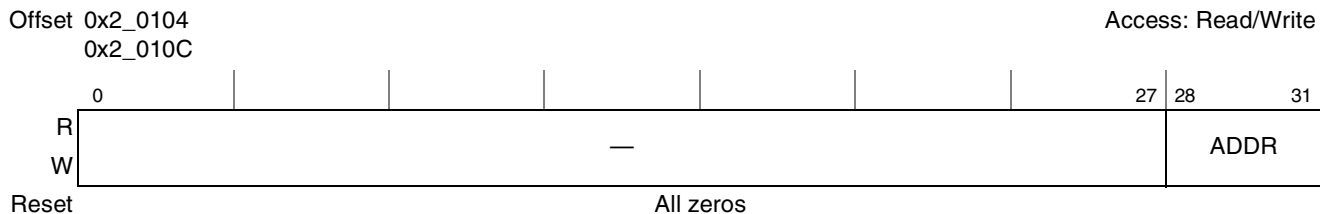
Bits	Name	Description												
0–17	ADDR	<p>Contains the lower 18 bits of the 22-bit L2 memory-mapped SRAM base address; the upper 4 bits are contained in L2SRBAREAn[ADDR]. (Note that some of these bits may not be needed, depending on how the L2 cache is partitioned.) The combined base address from L2SRBAREAn[ADDR]    L2SRBARn[ADDR] is used as follows:</p> <table border="1"> <thead> <tr> <th>SRAM Partition</th> <th>Bits Required for SRAM Offset</th> <th>Bits Used for Actual Base Address</th> </tr> </thead> <tbody> <tr> <td>64 Kbytes</td> <td>16</td> <td>20 (0–19)</td> </tr> <tr> <td>128 Kbytes</td> <td>17</td> <td>19 (0–18)</td> </tr> <tr> <td>256 Kbytes</td> <td>18</td> <td>18 (0–17)</td> </tr> </tbody> </table> <p>Unused bits of the base address are masked off by the hardware.</p>	SRAM Partition	Bits Required for SRAM Offset	Bits Used for Actual Base Address	64 Kbytes	16	20 (0–19)	128 Kbytes	17	19 (0–18)	256 Kbytes	18	18 (0–17)
SRAM Partition	Bits Required for SRAM Offset	Bits Used for Actual Base Address												
64 Kbytes	16	20 (0–19)												
128 Kbytes	17	19 (0–18)												
256 Kbytes	18	18 (0–17)												
18–31	—	Reserved												

When enabled, the windows defined in L2SRBAR $n$  and L2SRBAREAn supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR (see Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register (CCSRBAR).”) Overlapping SRAM and local access windows is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if SRAM and local access windows overlap.



### 7.3.1.3.2 L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREAn)

The L2 memory-mapped SRAM base address registers extended address (L2SRBAREAn), shown in Figure 7-12, contain the upper 4 bits of the L2 cache SRAM base address.



**Figure 7-12. L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREAn)**

Table 7-9 describes the fields of L2SRBAREAn.

**Table 7-9. L2SRBAREAn Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Contains the upper 4 bits of the L2 cache SRAM base address. Note that the 18 low-order bits of the base address are contained in L2SRBARn[ADDR].

### 7.3.1.4 L2 Error Registers

L2 error detection, reporting, and injection allow flexible handling of ECC and parity errors in the L2 data and tag arrays. When the device detects an L2 error, the appropriate bit in the error detect register (L2ERRDET) is set. Error detection is disabled by setting the corresponding bit in the error disable register (L2ERRDIS).

The address and attributes of the first detected error are also saved in the error capture registers (L2ERRADDR, L2ERRATTR, L2CAPTDATAHI, L2CAPTDATALO, and L2CAPTACC). Subsequent errors set error bits in the error detection registers, but information is saved only for the first one. Error reporting (by generating an interrupt) is enabled by setting the corresponding bit in the error interrupt enable register (L2ERRINTEN). Note that the error detect bit is set regardless of the state of the interrupt enable bit. When an error is detected, if error detection is enabled the L2 cache/SRAM always asserts an internal error signal with read data to prevent the L1 caches and architectural registers from being loaded with corrupt data. If error detection is disabled, the detected error bit is not set and no internal signal is asserted.

The L2 error detect register (L2ERRDET) is implemented as a bit-reset type register. Reading from this register occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0x0200\_0000 is written to the register.

Note that in SRAM mode, if a non-cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non-cache-line write transaction cannot be allocated in the L2.

### 7.3.1.4.1 Error Injection Registers

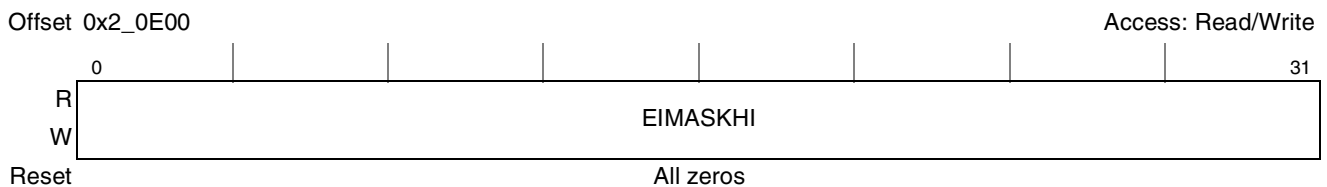
The L2 cache includes support for injecting errors into the L2 data, data ECC, or tag. This may be used to test error recovery software by deterministically creating error scenarios.

The preferred method for error injection is to set all data pages to cache-inhibited (MMU TLB entry I = 1) except a scratch page, set L2CTL[L2DO] to prevent allocation of instruction accesses, and invalidate the L2 by setting L2CTL[L2I] = 1. The following code sequence triggers an error, then detects it (A is an address in the scratch page):

```
dcbz A      | allocates the line in the L1 in the modified state
dcbt1s_L2 A | forces the line from the L1 and allocates the line in the L2
lwz A
```

Data or tag errors are injected into the line, according to the error injection settings in L2ERRINJHI, L2ERRINJLO, and L2ERRINJCTL, at allocation. The final load detects and reports the error (if enabled) and allows software to examine the offending data, address, and attributes.

Note that error injection enable bits in L2ERRINJCTL must be cleared by software and the L2 must be invalidated (by setting L2CTL[L2I]) before resuming L2 normal operation. [Figure 7-13](#) shows the L2 error injection mask high register (L2ERRINJHI).



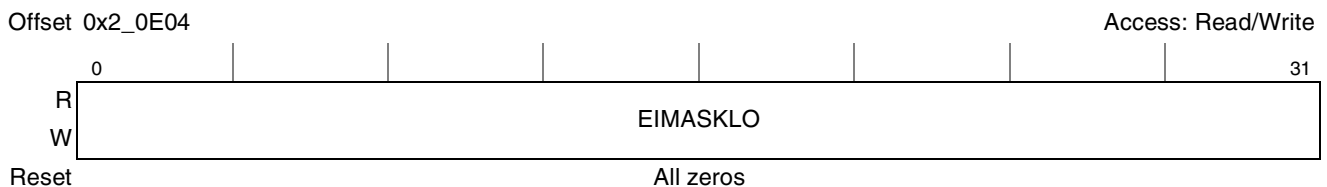
**Figure 7-13. L2 Error Injection Mask High Register (L2ERRINJHI)**

[Table 7-10](#) describes L2ERRINJHI[EIMASKHI].

**Table 7-10. L2ERRINJHI Field Description**

Bits	Name	Description
0–31	EIMASKHI	Error injection mask/high word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache/SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

[Figure 7-14](#) shows the L2 error injection mask low register (L2ERRINJLO).



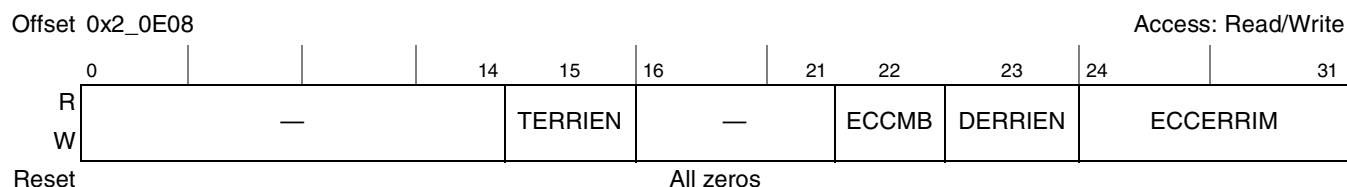
**Figure 7-14. L2 Error Injection Mask Low Register (L2ERRINJLO)**

Table 7-11 describes L2ERRINJLO[EIMASKLO].

**Table 7-11. L2ERRINJLO Field Description**

Bits	Name	Description
0–31	EIMASKLO	Error injection mask/low word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-15 shows the L2 error injection mask control register (L2ERRINJCTL).



**Figure 7-15. L2 Error Injection Mask Control Register (L2ERRINJCTL)**

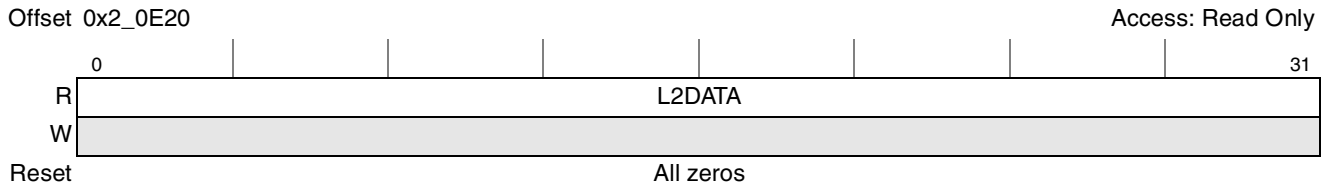
Table 7-12 describes L2ERRINJCTL fields.

**Table 7-12. L2ERRINJCTL Field Descriptions**

Bits	Name	Description
0–14	—	Reserved
15	TERRIEN	L2 tag array error injection enable 0 No tag errors are injected. 1 All subsequent entries written to the L2 tag array have the parity bit inverted.
16–21	—	Reserved
22	ECCMB	ECC mirror byte enable. 0 ECC byte mirroring is disabled 1 The most significant data path byte is mirrored onto the ECC byte if DERRIEN = 1.
23	DERRIEN	L2 data array error injection enable: 0 No data errors are injected. 1 Subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data and ECC error injection masks and/or data path byte mirrored onto ECC as specified by ECC mirror byte enable. Note: if both ECC mirror byte and data error injection are enabled, ECC mask error injection is performed on the mirrored ECC.
24–31	ECCERRIM	Error injection mask for the ECC bits. A set bit corresponding to an ECC bit causes that bit to be inverted on SRAM writes if DERRIEN = 1.

### 7.3.1.4.2 Error Control and Capture Registers

The error control and capture registers control detection and reporting of tag parity, ECC and L2 configuration errors. L2 configuration errors are illegal combinations of L2 size and block size and are detected when the L2 is enabled (L2CTL[L2E] = 1). Figure 7-16 shows the L2 error capture data high register (L2CAPTDATAHI).



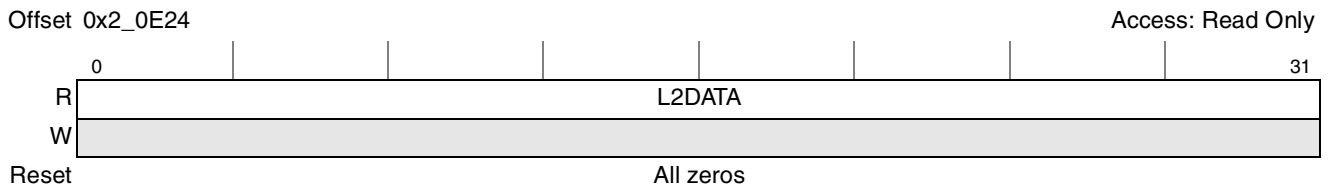
**Figure 7-16. L2 Error Capture Data High Register (L2CAPTDATAHI)**

Table 7-13 describes L2CAPTDATAHI[L2DATA].

**Table 7-13. L2CAPTDATAHI Field Description**

Bits	Name	Description
0–31	L2DATA	L2 data high word

Figure 7-17 shows the L2 error capture data low register (L2CAPTDATALO).



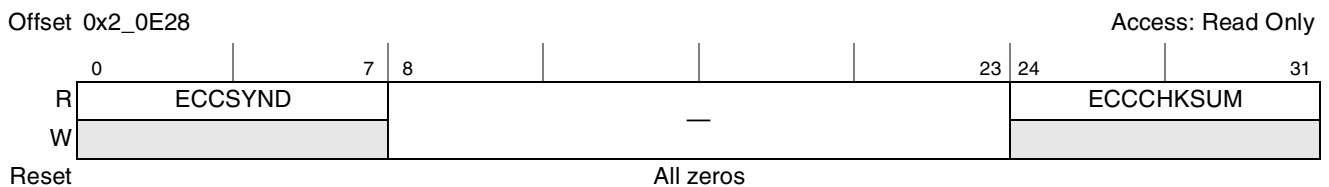
**Figure 7-17. L2 Error Capture Data Low Register (L2CAPTDATALO)**

Table 7-14 describes L2CAPTDATALO[L2DATA].

**Table 7-14. L2CAPTDATALO Field Description**

Bits	Name	Description
0–31	L2DATA	L2 data low word

Figure 7-18 shows the L2 error syndrome register (L2CAPTECC).



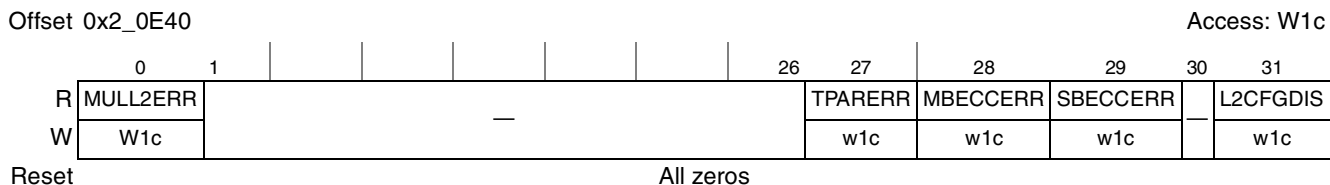
**Figure 7-18. L2 Error Syndrome Register (L2CAPTECC)**

Table 7-15 describes L2CAPTECC fields.

**Table 7-15. L2CAPTECC Field Descriptions**

Bits	Name	Description
0–7	ECCSYND	The calculated ECC syndrome of the failing double word
8–23	—	Reserved
24–31	ECCCHKSUM	The data path ECC of the failing double word

Figure 7-19 shows the L2 error detect register (L2ERRDET).



**Figure 7-19. L2 Error Detect Register (L2ERRDET)**

Table 7-16 describes L2ERRDET fields.

**Table 7-16. L2ERRDET Field Descriptions**

Bits	Name	Description
0	MULL2ERR	Multiple L2 errors (bit reset, write 1 to clear) 0 Multiple L2 errors of the same type were not detected 1 Multiple L2 errors of the same type were detected
1–26	—	Reserved
27	TPARERR	Tag parity error (bit reset, write 1 to clear) 0 Tag parity error was not detected 1 Tag parity error was detected Note that if an L2 cache tag parity error occurs on an attempt to write a new line, the L2 cache must be Flash invalidated. L2 functionality is not guaranteed if Flash invalidation is not performed after a tag parity error.
28	MBECCERR	Multiple-bit ECC error (bit reset, write 1 to clear) 0 Multiple-bit ECC errors were not detected 1 Multiple-bit ECC errors were detected
29	SBECCERR	Single-bit ECC error (bit reset, write 1 to clear) 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected.
30	—	Reserved
31	L2CFGERR	L2 configuration error (bit reset, write 1 to clear) 0 L2 configuration errors were not detected 1 L2 illegal configuration error detected. Reports inconsistencies between the L2SRAM, L2STASHDIS and L2STASHCTL fields of the L2 control register (L2CTL)

Figure 7-20 shows the L2 error disable register (L2ERRDIS).

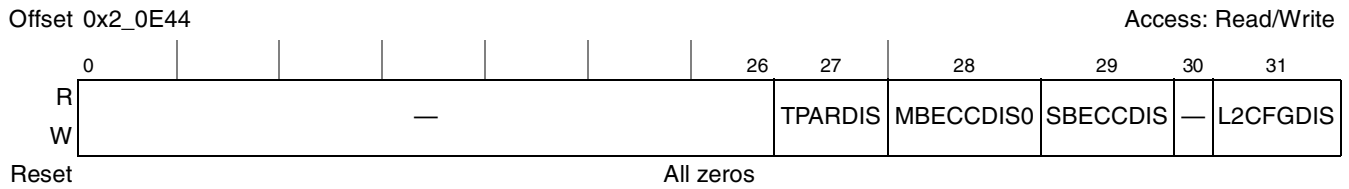


Figure 7-20. L2 Error Disable Register (L2ERRDIS)

Table 7-17 describes L2ERRDIS fields.

Table 7-17. L2ERRDIS Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARDIS	Tag parity error disable 0 Tag parity error detection enabled 1 Tag parity error detection disabled
28	MBECCDIS	Multiple-bit ECC error disable. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBECCDIS must be cleared and L2ERRINTEN[MBECCINTEN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 Multiple-bit ECC error detection enabled 1 Multiple-bit ECC error detection disabled
29	SBECCDIS	Single-bit ECC error disable 0 Single-bit ECC error detection enabled 1 Single-bit ECC error detection disabled
30	—	Reserved
31	L2CFGDIS	L2 configuration error disable 0 L2 configuration error detection enabled 1 L2 configuration error detection disabled

Figure 7-21 shows the L2 error interrupt enable register (L2ERRINTEN). When an enabled error condition exists, the L2 signals an interrupt to the core through the internal *int* signal.

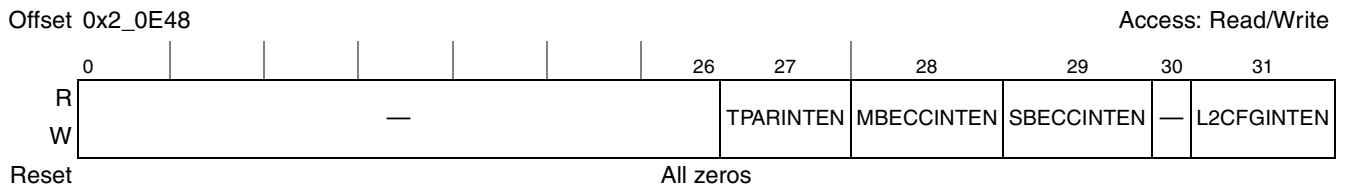


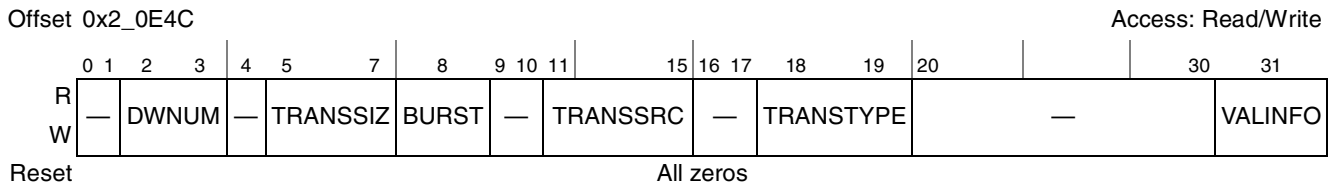
Figure 7-21. L2 Error Interrupt Enable Register (L2ERRINTEN)

Table 7-18 describes L2ERRINTEN fields.

**Table 7-18. L2ERRINTEN Field Descriptions**

Bits	Name	Description
0–26	—	Reserved
27	TPARINTEN	Tag parity error reporting enable 0 Tag parity error reporting disabled 1 Tag parity error reporting enabled
28	MBECCINTEN	Multiple-bit ECC error reporting enable. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, L2ERRDIS[MBECCDIS] must be cleared and MBECCINTEN must be set to ensure that an interrupt is generated. For more information, see <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1)”</a> . 0 Multiple-bit ECC error reporting disabled 1 Multiple-bit ECC error reporting enabled
29	SBECCINTEN	Single-bit ECC error reporting enable 0 Single-bit ECC error reporting disabled 1 Single-bit ECC error reporting enabled
30	—	Reserved
31	L2CFGINTEN	L2 configuration error reporting enable 0 L2 configuration error reporting disabled 1 L2 configuration error reporting enabled

Figure 7-22 shows the L2 error attributes capture register (L2ERRATTR).



**Figure 7-22. L2 Error Attributes Capture Register (L2ERRATTR)**

Table 7-19 describes L2ERRATTR fields.

**Table 7-19. L2ERRATTR Field Descriptions**

Bits	Name	Description																														
0–1	—	Reserved																														
2–3	DWNUM	Double-word number of the detected error (data ECC errors only)																														
4	—	Reserved																														
5–7	TRANSSIZ	Transaction size for detected error <table style="width: 100%; border: none;"> <tr> <td style="width: 30%;"></td> <td style="width: 30%; text-align: center;">Single-beat</td> <td style="width: 30%; text-align: center;">Burst</td> <td style="width: 30%;"></td> <td style="width: 30%; text-align: center;">Single-beat</td> <td style="width: 30%; text-align: center;">Burst</td> </tr> <tr> <td>000</td> <td>8 bytes</td> <td>Reserved</td> <td>100</td> <td>4 bytes</td> <td>Reserved</td> </tr> <tr> <td>001</td> <td>1 byte</td> <td>16 bytes</td> <td>101</td> <td>5 bytes</td> <td>Reserved</td> </tr> <tr> <td>010</td> <td>2 bytes</td> <td>32 bytes</td> <td>110</td> <td>6 bytes</td> <td>Reserved</td> </tr> <tr> <td>011</td> <td>3 bytes</td> <td>Reserved</td> <td>111</td> <td>7 bytes</td> <td>Reserved</td> </tr> </table>		Single-beat	Burst		Single-beat	Burst	000	8 bytes	Reserved	100	4 bytes	Reserved	001	1 byte	16 bytes	101	5 bytes	Reserved	010	2 bytes	32 bytes	110	6 bytes	Reserved	011	3 bytes	Reserved	111	7 bytes	Reserved
	Single-beat	Burst		Single-beat	Burst																											
000	8 bytes	Reserved	100	4 bytes	Reserved																											
001	1 byte	16 bytes	101	5 bytes	Reserved																											
010	2 bytes	32 bytes	110	6 bytes	Reserved																											
011	3 bytes	Reserved	111	7 bytes	Reserved																											

**Table 7-19. L2ERRATTR Field Descriptions (continued)**

Bits	Name	Description
8	BURST	Burst transaction for detected error 0 Single-beat ( $\leq 64$ bits) transaction 1 Burst transaction
9–10	—	Reserved
11–15	TRANSSRC	Transaction source for detected error 00000 External (system logic) 10000 Processor (instruction) 10001 Processor (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error 00 Snoop (tag/status read) 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VALINFO	L2 capture registers valid 0 L2 capture registers contain no valid information or no enabled errors were detected. 1 L2 capture registers contain information of the first detected error which has reporting enabled. Software must clear this bit to unfreeze error capture so error detection hardware can overwrite the capture address/data/attributes for a newly detected error.

Figure 7-23 shows the L2 error address capture register high (L2ERRADDRH).

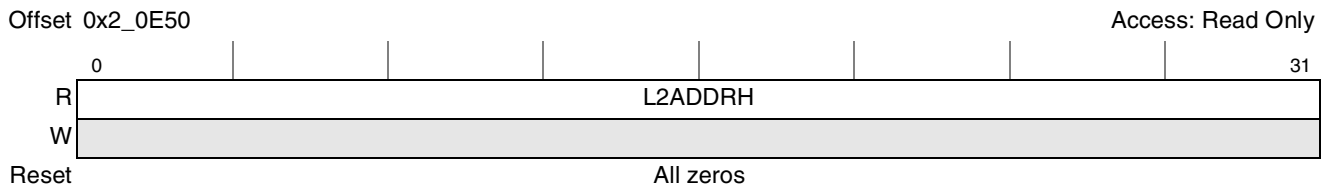
**Figure 7-23. L2 Error Address Capture Register (L2ERRADDRH)**

Table 7-20 describes L2ERRADDRH[L2ADDRH].

**Table 7-20. L2ERRADDRH Field Description**

Bits	Name	Description
0–31	L2ADDRH	L2 address bits 4–35 corresponding to detected error



Figure 7-24 shows the L2 error address capture register low (L2ERRADDRL).

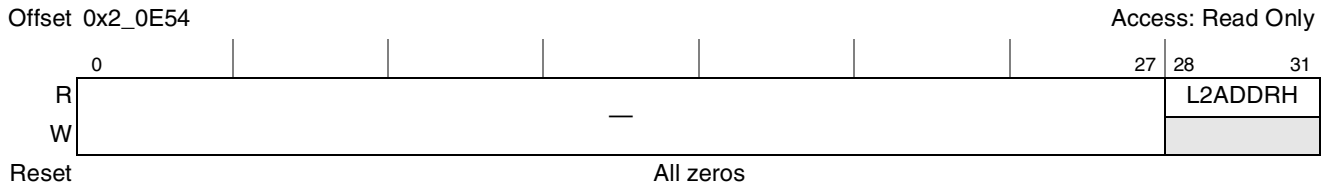


Figure 7-24. L2 Error Address Capture Register (L2ERRADDRL)

Table 7-21 describes L2ERRADDRL[L2ADDRH].

Table 7-21. L2ERRADDRL Field Description

Bits	Name	Description
0–27	—	Reserved
28–31	L2ADDRH	L2 address bits 0–3 corresponding to detected error

Figure 7-25 shows the L2 error control register (L2ERRCTL).

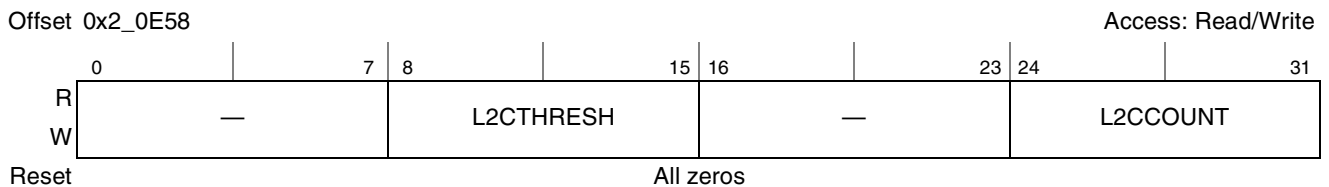


Figure 7-25. L2 Error Control Register (L2ERRCTL)

Table 7-22 describes L2ERRCTL fields.

Table 7-22. L2ERRCTL Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	L2CTHRESH	L2 cache threshold. Threshold value for the number of ECC single-bit errors that are detected before reporting an error condition.
16–23	—	Reserved
24–31	L2CCOUNT	L2 count. Counts ECC single-bit errors detected. If L2CCOUNT equals the ECC single-bit error trigger threshold, an error is reported if single-bit error reporting is enabled.

## 7.4 External Writes to the L2 Cache (Cache Stashing)

Data from an I/O master can be allocated into the L2 cache while simultaneously being written to memory. External (stashed) writes can be performed from any I/O master. For example:

- Ethernet
- PCI/PCI-Express
- DMA

Stashing is controlled either by an attribute from the initiator of a write or by address range registers in the L2 cache. New cache lines are allocated for full-cache-line writes (unless the line is already resident in the cache). Sub-cache-line write data is stashed only if the line is already valid in the cache. For these sub-cache-line writes, a read-modify-write process is used to merge the write data with the valid data already in the cache.

For information on how to initiate cache stashing from an I/O master, see the respective chapters for the I/O masters that support stashing.

For address range based control of stashing, the L2 cache external write address registers 0–3 (L2CEWAR $n$ ) and the L2 cache external write address registers extended address 0–3 (L2CEWAREA $n$ ) are used with the L2 cache external write control registers 0–3 (L2CEWCR $n$ ) to control the cache stashing functionality. Each register set (for example L2CEWAR0, L2CEWAREA0, and L2CEWCR0) specifies a programmed memory range that can be allocated and optionally locked with a global write transaction. The address register must be naturally aligned to the window size in the corresponding control register. For more information, see [Section 7.3.1.2, “L2 Cache External Write Registers.”](#)

Note that stashing can occur regardless of whether the L1 cache is enabled or whether the cache-inhibited bit in the MMU is set for the page.

## 7.4.1 Stash-Only Cache Regions

In order to prevent stashed I/O data from polluting processor data in the L2 cache (and vice versa), it is possible to create stash-only regions. This is controlled by the L2STASHCTL field of L2CTL. See [Section 7.3.1.1, “L2 Control Register \(L2CTL\).”](#)

If a stash-only region is created, then that region of the cache will only be used for stashed I/O data, and stashed I/O data will not cause the eviction of processor data; they are kept in separate ways of each set. The processor may allocate data into the ways of the cache that are not allocated to SRAM or stash-only memory. Replacement within the stash-only region and the processor region will be governed by a pseudo-LRU algorithm modified by masks that allow only applicable ways of a cache set to be considered for replacement.

## 7.5 L2 Cache Timing

Table 7-23 shows the timing of back-to-back loads that miss in the L1 data cache and hit in the L2 cache, assuming the core is running at 2 1/2 times the L2 cache frequency. The L2 returns the 128 bits containing the requested data (critical quad word) first. This data is forwarded to the result register before the full cache line reloads the L1.

**Table 7-23. Fastest Read Timing—Hit in L2**

Core Clocks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
e500 core load 1	to D-cache	D-cache miss	to CIU	CIU Q												to CIU	LSU DLFb	LSU reads command	LSU reads out data	Result bus							
e500 core load 2		to D-cache	D-cache miss	to CIU	CIU Q																to CIU	LSU DLFb	LSU reads command	LSU reads out data	Result bus		
CCB clocks	<1	2		3		4		5		6		7		8		9		10		11		12		13		14	
CCB addr bus load 1		BG		TS				AACK		HIT DATA-COMING																	
CCB addr bus load 2						BG		TS				AACK		HIT DATA-COMING													
CCB data bus load 1												DATA		DATA													
CCB data bus load 2																		DATA		DATA							

## 7.6 L2 Cache and SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term ‘snoop transaction’ refers to transactions initiated by the system logic or by I/O traffic, as opposed to e500 core-initiated transactions.

## 7.6.1 L2 Cache Coherency Rules

L2 cache coherency rules are as follows:

- The L2 is non-inclusive of the L1—valid L1 lines may be valid or invalid in the L2.
- The L2 cache holds no modified data. Data is in one of four states—invalid, exclusive, exclusive locked, and stale.
- The L2 allocates entries for data cast out or pushed (non-global, non-write-through write with kill) from the L1 caches.
- Lines for e500 core-initiated burst read transactions are allocated as exclusive in the L2.
- The L2 supports I/O devices reading data from valid lines in the L2 cache (data intervention) if  $L2CTL[L2INTDIS] = 0$ . An optional unlock attribute causes I/O reads to clear a lock when the read is performed.
- The L2 cache does not respond to cache-inhibited read transactions.
- e500 core-initiated, cache-inhibited store transactions invalidate the line when they hit on a valid L2 line. If the line is locked, it goes to the stale state. For other write transactions the cache-inhibited bit is ignored.
- Non-burst cacheable write transactions from the e500 core (generated by write-through cacheable stores) update a valid L2 cache line through a read-modify-write operation.
- e500 core cast out transactions that hit on a stale line in the L2 cache cause a data update of the line and a change to the valid locked state for that line.
- An e500 core-initiated, cacheable, non-write-through store that misses in the L1 and hits on a line in the L2 invalidates that line in the L2. If the line is marked exclusive locked, the L2 marks the line as stale.
- Transactions that hit a stale L2 cache line that would cause an allocate if they miss cause a data update of the line (when data arrives from memory) and a change to the line's valid locked state. Data is not supplied by the L2 cache for the read in this case.
- The following transactions kill the data and the respective locks when they hit a valid L2 line:
  - **dcbf**
  - **dcbi**
- The L2 cache supports mixed cache external writes and core-initiated writes to the same addresses if the core-initiated writes are marked coherency-required, caching allowed, not write-through ( $WIMG = 001x$ ) and the external writes are marked coherency-required, caching-allowed.
- The L2 cache supports writes to the L2 cache from peripheral devices or from I/O controllers through snoop write transactions with addresses that hit in a programmed memory range. Full cache line (32-byte) write transactions update the data for a valid line in the L2 and if the line is not valid in the L2, a line is allocated. Sub-cache line write transactions update the data only for valid L2 cache lines through read-modify-write operations.
- The L2 cache supports burst writes that lock an L2 cache line from peripheral devices or from I/O controllers through write transactions with addresses that hit in a programmed memory range that has the lock attribute set.

- The L2 cache supports burst writes that allocate and/or lock an L2 cache line from peripheral devices or I/O controllers through a write allocate transaction. See the system logic programming model (for example, that of the DMA controller) for details on how to set the transaction type for cache external writes to the L2.

## 7.6.2 Memory-Mapped SRAM Coherency Rules

Memory-mapped SRAM coherency rules are as follows:

- External (non-core-initiated) accesses to memory-mapped SRAM must be marked coherency-required. External accesses to memory-mapped SRAM marked coherency-not-required may cause an address unavailable error.
- Accesses to memory-mapped SRAM are cacheable only in the corresponding e500 L1 caches. External accesses must be marked cache-inhibited or be performed with non-caching transactions.

## 7.7 L2 Cache Locking

The caches can be locked and cleared using the following methods:

- Cache locking methods
  - Individual line locks are set and cleared by using instructions defined by the e500 cache locking APU, which is part of the Freescale Embedded Implementation Standards (EIS). These instructions include Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**). For detailed information about these instructions, see the *PowerPC e500 Core Reference Manual*.
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR<sub>n</sub>).
  - The entire cache can be locked by setting a configuration registers appropriately
- Methods for clearing locks
  - Individual locks can be cleared by cache locking APU instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless the entire cache is locked.
  - Flash clearing of all instruction and/or data locks can be done by writes to configuration registers.
  - An unlock attribute can be attached to I/O read operations.

### 7.7.1 Locking the Entire L2 Cache

The entire L2 cache can be locked by setting L2CTL[L2DO] = 1 and L2CTL[L2IO] = 1. This has the effect of preventing any further allocation of new lines in the cache by core requests. If there are lines in the cache that are not valid, they cannot be used by core requests until the cache is unlocked. While the cache is locked, read requests are serviced as normal, and snooping continues as normal to maintain

coherency. Lines invalidated to satisfy coherency requirements cannot be reallocated by core requests while the cache remains locked. The L2 cache can be unlocked by clearing L2CTL[L2IO] and/or L2CTL[L2DO]. Note that L2CTL[L2DO] and L2CTL[L2IO] have no effect on cache external write allocations or memory-mapped SRAM.

Note that this form of cache locking does not use the lock bits of the cache and cannot be cleared by resetting the cache or lock bits.

## 7.7.2 Locking Programmed Memory Ranges

A programmed memory range can be locked with a snoop write transaction that matches a cache external write address range (specified by L2CEWAR<sub>n</sub>/L2CEWAREA<sub>n</sub> and L2CEWCR<sub>n</sub>). There are no clearing of locks through the programmed address ranges. Locks can be cleared using clear lock instructions, flushes, read-and-clear-lock snoop (RWNITC with clear lock attribute), or flash clear locks.

## 7.7.3 Locking Selected Lines

Individual lines are locked when the L2 receives one of the following burst transactions:

- **icbtls** (CT = 1)—Instruction Cache Block Touch and Lock Set instruction
- **dcbtls** (CT = 1)—Data Cache Block Touch and Lock Set instruction
- **dcbtstls** (CT = 1)—Data Cache Block Touch for Store and Lock Set instruction
- Snoop burst write—If the address hits on a programmed cache external write space with the lock attribute set, or if the write allocate transaction type is used
- Snoop non-burst write—If the address hits on a programmed cache external write space with the lock attribute set

Note that the core complex broadcasts these instructions to the L2 if the CT field in the instruction specifies the L2 cache (CT = 1). When the L2 cache is specified, data is not placed in the L1, only the L2. If the L1 cache is specified (CT = 0), the L2 does not lock the line, and the data is placed in the L1 (and locked).

When the touch lock set L2 instruction (**dcbtls** or **dcbtstls**) hits are modified in the L1 cache, the modified data is allocated into the L2 cache (and written back to main memory) and a data lock is set. The L1 line state transitions to invalid.

Note that if the L2 receives a request to allocate and lock a line, but all lines in the selected way are locked, the requested L2 line is not allocated and the L2 cache lock overflow bit (L2CTL[L2LO]) is set.

Lines invalidated to satisfy coherency requirements cannot be reallocated while the cache remains locked.

## 7.7.4 Clearing Locks on Selected Lines

Individual locks in the L2 are cleared by a lock clear (**icblc** or **dcblc**, CT = 1) instruction. This directs the L2 cache to clear a lock on that line if it hits in the L2 cache. Both data and instruction locks are cleared by the **icblc** and **dcblc** instructions.

Note that the lock on a line is cleared if the line is invalidated by a snooped Flush transaction, and the line in the cache is available for allocation of a new line of instruction or data unless the entire cache is locked.

## 7.7.5 Flash Clearing of Instruction and Data Locks

Locks for instructions and data are recorded separately in the L2 cache, and they can be flash cleared separately by writing the appropriate value to the L2 cache control register (L2CTL[L2LFR] and L2CTL[L2LFRID]). Flash invalidating of the L2 (setting L2CTL[L2I]) clears all locks on both instructions and data.

Note that flash clearing is the only way to clear data locks without clearing instruction locks, or to clear instruction locks without clearing data locks. All instructions and snoop transactions that clear locks clear both data and instruction locks.

## 7.7.6 Locks with Stale Data

If data is locked in the L2 and either the e500 core performs a cacheable copyback store or a **dcbtst** misses in the L1, the L2 invalidates the line; however, the L2 clears the valid bit for the data, the lock remains, and the line cannot be victimized. If the e500 core casts out modified data or pushes it in response to a non-flush snoop, the L2 updates the data and sets the valid bit again, maintaining the lock and keeping the data in the cache hierarchy.

## 7.8 PLRU L2 Replacement Policy

Line replacement is determined using a pseudo least-recently-used (PLRU) algorithm. There is a valid bit (V0–V7) for each line. To determine the replacement victim (the line to be cast out), there are seven PLRU bits (P0–P6) for each set. PLRU bits are updated every time a new line is allocated and every time an existing line is read by the processor, updated by a write, or invalidated.

Figure 7-26 shows the binary decision tree used to generate the victim line. The eight ways of the L2 cache are labeled W0–W7; the seven PLRU bits are labeled P0–P6.

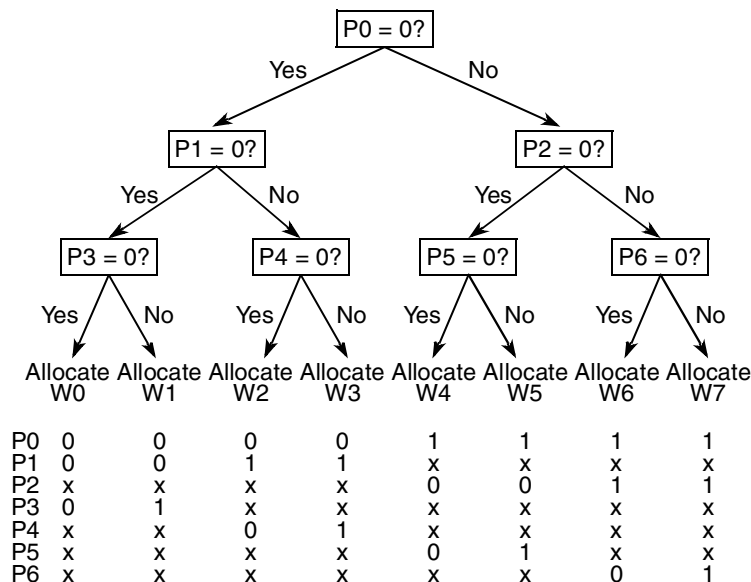


Figure 7-26. L2 Cache Line Replacement Algorithm

## 7.8.1 PLRU Bit Update Considerations

PLRU bit updates depend on which cache way was last accessed, as summarized in [Table 7-24](#).

**Table 7-24. PLRU Bit Update Algorithm**

Last Way Accessed	PLRU Bits						
	P0	P1	P2	P3	P4	P5	P6
0	1	1	—	1	—	—	—
1	1	1	—	0	—	—	—
2	1	0	—	—	1	—	—
3	1	0	—	—	0	—	—
4	0	—	1	—	—	1	—
5	0	—	1	—	—	0	—
6	0	—	0	—	—	—	1
7	0	—	0	—	—	—	0

When an L2 line is invalidated, the PLRU bits are updated, marking the corresponding way as least-recently used. This causes the invalidated way to be selected as the next victim.

## 7.8.2 Allocation of Lines

The general PLRU algorithm described above must be modified to take into account special features of the L2 cache; namely SRAM regions, line locking, and stash-only regions. Each of these features reserves ways within each cache set such that some ways are not eligible for allocation/victimization by the general LRU algorithm.

To preserve the state of the ways that are set aside for other special functions, the PLRU pointers are modified by a mask that is a function of the L2 configuration registers, the lock bits in the cache status array, and initiator of the transaction. The mask effectively points the PLRU algorithm away from ways that are not to be considered for replacement.

L2 cache lines are locked through the status array lock bits. There are two lock bits for each way of each set (1024 sets by eight ways). These bits are set or cleared through special L2 controller commands. There are two sets of lock bits, one for instructions (I0–I7) and one for data (D0–D7) for every line. The lock bits act as a mask over the PLRU bits to determine victim selection. The PLRU bits are updated regardless of line locking.

Lock bits are used at allocate time to steer the PLRU algorithm away from selecting locked victims. In the following discussion, the eight lock bits for a particular set are called L0–L7.

Where Lock Way  $i$ :  $L_i = D_i \mid I_i$ ,  $i=0..7$  ( $D_i$  = data lock,  $I_i$  = instruction lock)



An effective value of each PLRU bit is calculated as follows:

$$\begin{aligned}
 P0\_eff &= f(P0, L0, L1, L2, L3, L4, L5, L6, L7) = (L0 \& L1 \& L2 \& L3) \mid (P0 \& \sim(L4 \& L5 \& L6 \& L7)) \\
 P1\_eff &= f(P1, L0, L1, L2, L3) = (L0 \& L1) \mid (P1 \& \sim(L2 \& L3)) \\
 P2\_eff &= f(P2, L4, L5, L6, L7) = (L4 \& L5) \mid (P2 \& \sim(L6 \& L7)) \\
 P3\_eff &= f(P3, L0, L1) = L0 \mid (P3 \& \sim L1) \\
 P4\_eff &= f(P4, L2, L3) = L2 \mid (P4 \& \sim L3) \\
 P5\_eff &= f(P5, L4, L5) = L4 \mid (P5 \& \sim L5) \\
 P6\_eff &= f(P6, L6, L7) = L6 \mid (P6 \& \sim L7)
 \end{aligned}$$

These effective PLRU bits are used to select a victim, as indicated in [Table 7-25](#).

**Table 7-25. PLRU-Based Victim Selection Mechanism**

Way Selected	Effective PLRU State (Binary)	Reduced Logic Equation (using effective PLRU bits)
W0	00x0xxx	$\sim P0 \& \sim P1 \& \sim P3$
W1	00x1xxx	$\sim P0 \& \sim P1 \& P3$
W2	01xx0xx	$\sim P0 \& P1 \& \sim P4$
W3	01xx1xx	$\sim P0 \& P1 \& P4$
W4	1x0xx0x	$P0 \& \sim P2 \& \sim P5$
W5	1x0xx1x	$P0 \& \sim P2 \& P5$
W6	1x1xxx0	$P0 \& P2 \& \sim P6$
W7	1x1xxx1	$P0 \& P2 \& P6$

## 7.9 L2 Cache Operation

This section describes the behavior of the L1 and L2 cache in response to various operations and in various configurations.

### 7.9.1 Initialization

#### 7.9.1.1 L2 Cache Initialization

After power-on reset the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a flash invalidate before using the array as an L2 cache. This is done by writing a one to the L2I field of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

#### 7.9.1.2 Memory-Mapped SRAM Initialization

After power-on reset the contents of the data and ECC arrays are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the processor or any other device that uses sub-cache-line transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cache-line writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error

disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See [Section 7.3.1.4.2, “Error Control and Capture Registers.”](#) If the array is initialized by a DMA engine using cache-line writes, ECC checking can remain enabled during the initialization process.

## 7.9.2 Flash Invalidation of the L2 Cache

The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core complex bus while the flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a flash invalidation of the L2 cache regions of the array.

## 7.9.3 Managing Errors

### 7.9.3.1 ECC Errors

An individual soft error that causes a single- or multi-bit ECC error can be cleared from the L2 array simply by performing a **dcbf** instruction on the address captured in the L2ERRADDR register. This will invalidate the line in the L2 cache. When the load that caused the ECC error is performed again, the data will be reallocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be flash invalidated to clear out all single-bit errors.

Note that no data is lost by **dcbfs** or flash invalidates, since the L2 cache is write-through and contains no modified data.

### 7.9.3.2 Tag Parity Errors

A tag parity error must be fixed by flash invalidating the L2 cache. Note that a **dcbf** operation to the address that caused the error to be reported is not sufficient since a tag parity error is seen as an L2 miss and will not cause invalidation of the bad tag. Proper L2 operation cannot be guaranteed if an L2 tag parity error is not repaired by a flash invalidation of the entire array.

## 7.9.4 L2 Cache States

The L2 status array uses four bits for each line to determine the status of the line. Different combinations of these bits result in different L2 states. The status bits are as follows:

- Valid (V)
- Instruction locked (IL)

- Data locked (DL)
- Stale (T)

Table 7-26 shows L2 cache states. Note that these conventions are also used in Table 7-27.

**Table 7-26. L2 Cache States**

V	T	IL	DL	L2 states
0	x	x	x	Invalid (I)
1	0	0	0	Exclusive (E)
1	0	0	1	Exclusive data locked (EDL)
1	0	1	0	Exclusive instruction locked (EIL)
1	0	1	1	Exclusive instruction and data locked (EL)
1	1	0	0	Stale (data invalid, locks invalid) (T)
1	1	0	1	Stale (data invalid, dlock valid) (TDL)
1	1	1	0	Stale (data invalid, ilock valid) (TIL)
1	1	1	1	Stale (data invalid, locks valid) (TL)

## 7.9.5 L2 State Transitions

Table 7-27 lists state transitions for all e500 core-initiated transactions that change the L2 cache state. Core-initiated transactions caused when the core executes **msync**, **mbar**, **tlbivax**, or **tlbsync** do not change the L2 cache state. The table does not list initial L1 states for transactions that hit in the L1 (iL1 or dL1) and are not sent to the L2.

In the table, the heading ‘L2 hit’ indicates that the L2 provides (on a read) or captures (on a write) data for an existing line. Some entries list two final L1 states. L2 touch instructions never allocate into iL1 or dL1.

Note that if the L2 SRAM is disabled, the L2 initial and final states are always I and the L2 never hits. Similarly, if the L2 SRAM is in full memory-mapped SRAM mode, the L2 initial and final states are always I and the L2 never hits for addresses not in the memory-mapped SRAM address range. The L2 always hits for addresses in the enabled memory-mapped SRAM address ranges.

**Table 7-27. State Transitions Due to Core-Initiated Transactions**

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable instruction fetch icbtis_L1	iL1 I	I/T	No	I/V	same	L2CTL[L2DO] = 1. L2 touch instructions not allocated in L1
		I	No	I/V	E	L2CTL[L2DO] = 0
icbt_L2	dL1 I,E	E/EL	Yes	I/V	same	
		T	No	I/V	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus

Table 7-27. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
icbtls_L2	dL1 I,E	I/T	No	I	same	L2CTL[L2DO] = 1
		E	Yes	I	I	L2CTL[L2DO] = 1
		EL	Yes	I	T	L2CTL[L2DO] = 1
		I	No	I	EL	L2CTL[L2DO] = 0
		E	Yes	I	EL	L2CTL[L2DO] = 0
		EL	Yes	I	same	L2CTL[L2DO] = 0
		T	No	I	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited instruction fetch	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cacheable load (4-state) Cacheable <b>lwarx</b> (4-state) dcbt_L1 (4-state) dcbtls_L1 (4-state)	dL1 I	I/T	No	E	same	L2CTL[L2IO] = 1
		E	Yes	E	I	L2CTL[L2IO] = 1
		EL	Yes	E	T	L2CTL[L2IO] = 1
		I	No	E	E	L2CTL[L2IO] = 0
		E/EL	Yes	E	same	L2CTL[L2IO] = 0
		T	No	EL	EL	L2CTL[L2IO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited load	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cache-inhibited <b>lwarx</b>	N/A	N/A	No	N/A	N/A	No L2 effect
Writeback Store	dL1 I	I/T	No	M	same	L2 allocates when a line is cast out of L1.
		E	Yes	M	I	
		EL	Yes	M	T	
Writeback <b>stwcx</b>	dL1 I	I/T	No	M	same	
		E	Yes	M	I	
		EL	Yes	M	T	
Cacheable load (3-state) Cacheable <b>lwarx</b> (3-state) dcbt_L1 (3-state) dcbtls_L1 (3-state)	dL1 I	I	No	E/I	I	L2CTL[L2IO] = 1
		T	No	E/I	T	L2CTL[L2IO] = 1
		E	Yes	E/I	I	L2CTL[L2IO] = 1
		EL	Yes	E/I	T	L2CTL[L2IO] = 1
		I	No	E/I	E	L2CTL[L2IO] = 0
dcbt_L2 dcbtst_L2	dL1 I,E	E/EL	Yes	E/I	same	L2CTL[L2IO] = 0
		T	No	E/I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
dcbtst_L1 dcbtstls_L1	dL1 I	I/T	No	E	same	
		E	Yes	E	I	
		EL	Yes	E	T	

Table 7-27. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbtIs_L2 dcbtstIs_L2	dL1 I,E	I	No	I	I	L2CTL[L2IO] = 1
		T	No	I	T	L2CTL[L2IO] = 1
		E	Yes	I	I	L2CTL[L2IO] = 1
		EL	Yes	I	T	L2CTL[L2IO] = 1
		I	No	I	EL	L2CTL[L2IO] = 0
		E/EL	Yes	I	EL	L2CTL[L2IO] = 0
		T	No	I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
Write-through store	dL1 I,E,M	I/T	No	same	I	
		E/EL	Yes	same	same	Read-modify-write
Cache-inhibited store	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
Cache-inhibited <b>stwcx</b>	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
dcbLc_L2 icbLc_L2	dL1 I,E,M	I/E	No	same	same	
		EL	No	same	E	
		T	No	same	I	
Victim castout dcbt_L2 icbt_L2 dcbtst_L2	dL1 M	I/T	No	I	same	L2CTL[L2IO] = 1. If software sharing cache lines between instructions and data wishes to capture instruction lines in L2 with L2CTL[L2IO] = 1, it must perform <b>dcbst</b> to flush the line out of the dL1 before fetching it into L2.
		I	No	I	E	L2CTL[L2IO] = 0
		E/EL	No	I	I/T	L2CTL[L2IO] = 1.
			Yes	I	Same	L2CTL[L2IO] = 0.
		T	Yes	I	EL	L2CTL[L2IO] = 0.
dcbtIs_L2 icbtIs_L2 dcbtstIs_L2	dL1 M	I	No	I	EL	An icbtIs_L2 that hits modified in L1 cannot be distinguished from dcbtIs_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set hillocks in L2, it must perform <b>dcbst</b> to flush the line out of the dL1 before locking it in L2.
		E/EL/T	Yes	I	EL	
Snoop push	dL1 M	I/E	No	I/E	I	
		EL/T	No	I/E	T	Invalidate data, keep lock
<b>dcbf</b> <b>dcbst</b>	dL1 M	I/E/EL	No	I	I	
<b>dcbz</b> <b>dcba</b>	dL1 I	I/E	No	M	I	
		EL	No	M	T	

Table 7-27. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbi	dL1 I,E,M	I/ E/EL/T	No	I	I	
dcbf dcbst	dL1 I,E	I/ E/EL/T	No	I	I	
icbi	iL1 I,V	I/ E/EL/T	No	I	I	

Table 7-28 lists L2 cache state transitions for all system-initiated (non-core) transactions that change the L2. The transaction types and attributes listed follow MPX bus nomenclature, with the addition of write allocate (burst write with L2 cache allocation). Table 7-28 accounts for changes caused by L1 snoop pushes triggered by snoops, listed in Table 7-27.

Table 7-28. State Transitions Due to System-Initiated Transactions

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Clean IKill	x	x	0	I/E/EL/T	Same	
Flush	x	x	0	I/E/EL/T	I	
Write allocate	x	1	0	I/E/EL/T	EL	Allocate and lock regardless of cache external write (CEW) window
				I/E	E	Allocate regardless of CEW window
	x	0	0	I/E	I	No allocate if cache-inhibited
				EL/T	T	Invalidate data, keep lock
WWK 32-byte WWF 32-byte WWF atomic	x	1	0	I/E/EL/T	I	Miss in cache external write windows
				I	E/EL	Hit in cache external write window
				EL	Same	Hit in cache external write window
				E	E/EL	Hit in cache external write window
				T	EL	Hit in cache external write window
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock
< 32-byte WWF < 32-byte WWF atomic	x	1	0	I/E	I	Miss in cache external write windows
				EL/T	T	Miss in cache external write windows.
				I/T	Same	Hit in CEW window but need burst data
				EL	Same	Hit in cache external write window
				E	E/EL	Hit in cache external write window. Set lock if CEW lock attribute set.
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock

Table 7-28. State Transitions Due to System-Initiated Transactions (continued)

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Read Read atomic	1	1	0	I/T	Same	
				E	E	
				EL	EL	
	x	0	0	N/A	N/A	No L1/L2 effect
RWNITC	1	1	0	I//T	Same	
				E	E	
				EL	EL	
	0	1	0	I	Same	Read-and-clear-lock
				EL	E	
				T	I	
	x	0	0	N/A	N/A	No L1/L2 effect
Kill RWITM RWITM atomic RClaim	x	1	0	I/E	I	
				EL/T	T	Invalidate data, keep lock
	x	0	0	I/E/EL/T	Same	

## 7.9.6 Error Checking and Correcting (ECC)

The L2 cache supports error checking and correcting (ECC) for the data path between the core master and system memory. It detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected.

Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but double- or multi-bit errors generate an interrupt.

The syndrome encodings for the ECC code are shown in [Table 7-29](#) and [Table 7-30](#).

Table 7-29. L2 Cache ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	

Table 7-29. L2 Cache ECC Syndrome Encoding (continued)

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
9	•		•				•		41			•		•		•	
10	•			•			•		42	•		•		•		•	•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•			•	•		
17		•		•				•	49			•		•	•		
18		•			•			•	50				•		•	•	
19	•	•			•				51	•				•	•		
20		•	•			•			52		•			•		•	
21		•		•		•			53			•		•		•	
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•				•		•	
24		•	•				•		56		•				•	•	
25		•		•			•		57			•			•	•	
26		•			•		•		58				•		•	•	
27	•	•			•		•	•	59	•					•	•	
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 7-30. L2 Cache ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•



## Part III

# Memory, Security, and I/O Interfaces

Part III defines the memory, security and I/O interfaces of the MPC8533E and it describes how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- [Chapter 8, “e500 Coherency Module,”](#) defines the e500 coherency module and how it facilitates communication between the e500 core complex, the L2 cache and the other blocks that comprise the coherent memory domain of the MPC8533E.

The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8533E.

- [Chapter 9, “DDR Memory Controller,”](#) describes the DDR SDRAM memory controller of the MPC8533E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. Special features like ECC error injection support rapid system debug.
- [Chapter 10, “Programmable Interrupt Controller,”](#) describes the embedded programmable interrupt controller (PIC) of the MPC8533E. This controller is an OpenPIC-compliant interrupt controller that provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.
- [Chapter 11, “I<sup>2</sup>C Interfaces,”](#) describes the two inter-IC (IIC or I<sup>2</sup>C) bus controllers of the MPC8533E. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8533E powers up in boot sequencer mode which allows the I<sup>2</sup>C1 controller to initialize configuration registers.
- [Chapter 12, “Security Engine \(SEC\) 2.1,”](#) describes the security controller of the MPC8533E.
- [Chapter 13, “DUART,”](#) describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- [Chapter 14, “Local Bus Controller,”](#) describes the local bus controller of the MPC8533E. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM),

SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- [Chapter 15, “Enhanced Three-Speed Ethernet Controllers,”](#) describes the two enhanced three-speed Ethernet controllers of the MPC8533E. These controllers provide 10/100/1Gb Ethernet support with a complete set of media-independent interface options including SGMII, GMII, RGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8533E memory coherency module.
- [Chapter 16, “DMA Controller,”](#) describes the four-channel general-purpose DMA controller of the MPC8533E. The DMA controller transfers blocks of data, independent of the e500 core or external hosts. Data movement occurs among the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- [Chapter 17, “PCI Bus Interface,”](#) describes the PCI controller of the MPC8533E.
- [Chapter 18, “PCI Express Interface Controller,”](#) describes the PCI-Express implementation of the MPC8533E.

# Chapter 8

## e500 Coherency Module

### 8.1 Introduction

The e500 coherency module (ECM) provides a flexible, easily expandable switching structure for routing e500- and I/O-initiated transactions to target modules on the device. Figure 8-1 shows a high-level block diagram of the ECM.

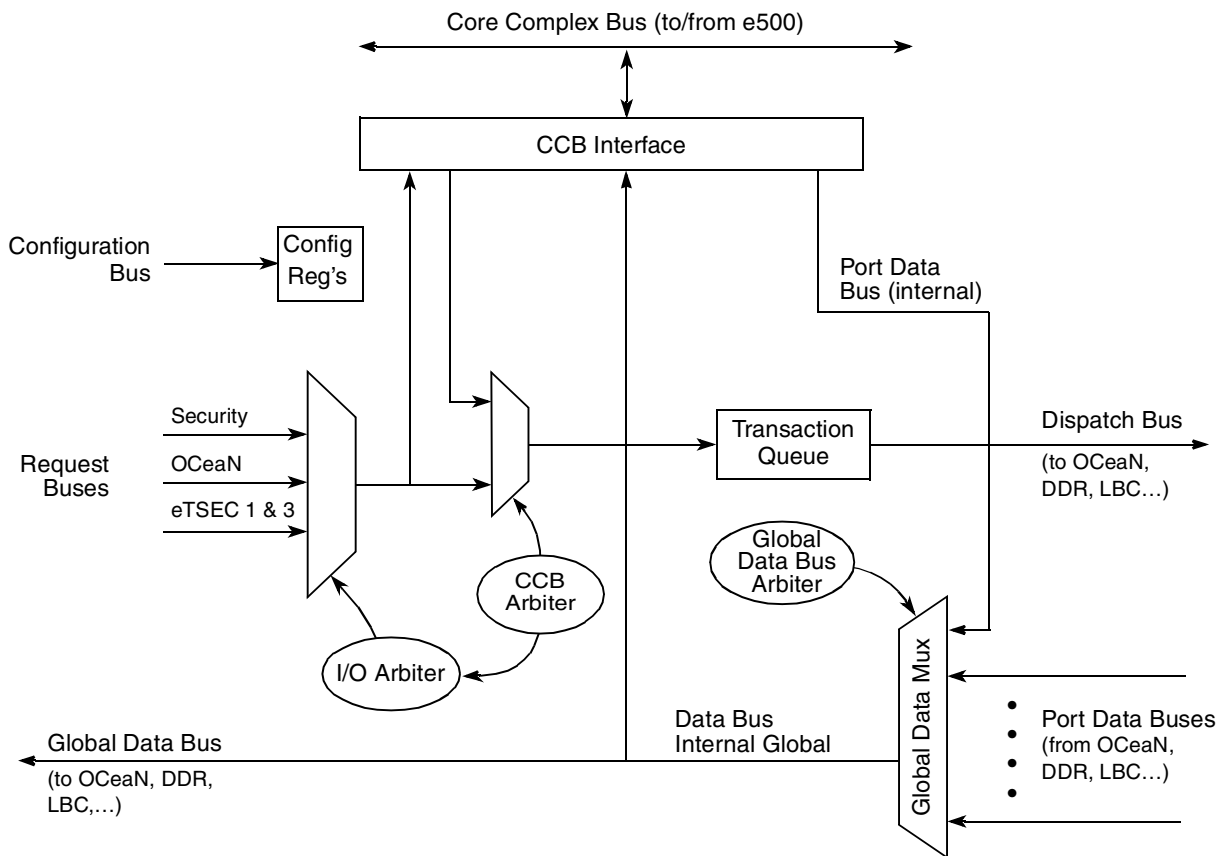


Figure 8-1. e500 Coherency Module Block Diagram

## 8.1.1 Overview

The ECM routes transactions initiated by the e500 core to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the ECM forwards I/O-initiated transactions that are tagged with the global attribute onto the core complex bus (CCB). This allows on-chip caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory.

## 8.1.2 Features

The ECM includes these distinctive features:

- Support for the e500 core and an L2/SRAM on the CCB, including a CCB arbiter.
- It sources a 64-bit data bus for returning read data from the ECM to the e500 core and routing write data from the ECM to the L2/SRAM. It sinks a 128-bit data bus for receiving data from the L2/SRAM and a 128-bit write data bus from the e500 core.
- Four connection points for I/O initiating (mastering into the device) interfaces. The ECM supports five connection points for I/O targets. The DDR memory controller, local bus, OCeaN targets, and configuration register access block all have a target port connection to the ECM.
- Split transaction support—separate address and data tenures allow for pipelining of transactions and out-of-order data tenures between initiators and targets.
- Proper ordering of I/O-initiated transactions.
- Speculative read bus for low-latency dispatch of reads to the DDR controller.
- Low-latency path for returning read data from DDR to the e500 core.
- Error registers trap transactions with invalid addresses. Errors can be programmed to generate interrupts to the e500 core, as described in the following sections:
  - [Section 8.2.1.5, “ECM Error Detect Register \(EEDR\)”](#)
  - [Section 8.2.1.6, “ECM Error Enable Register \(EER\)”](#)
  - [Section 8.2.1.7, “ECM Error Attributes Capture Register \(EEATR\)”](#)
  - [Section 8.2.1.8, “ECM Error Low Address Capture Register \(EELADR\)”](#)
  - [Section 8.2.1.9, “ECM Error High Address Capture Register \(EEHADR\)”](#)
- Errors from reading I/O devices (for example a master-aborted read transaction on the PCI interface) terminate with data sent to the master with a corrupt attribute. If the master is the e500 core, the ECM asserts *core\_fault\_in* to the core, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and one of these errors occurs, appropriate interrupts must be enabled to ensure that an interrupt is generated. See [Section 6.10.2, “Hardware Implementation-Dependent Register 1 \(HID1\).”](#)

## 8.2 Memory Map/Register Definition

Table 8-1 shows the ECM's memory map. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 8-1. ECM Memory Map**

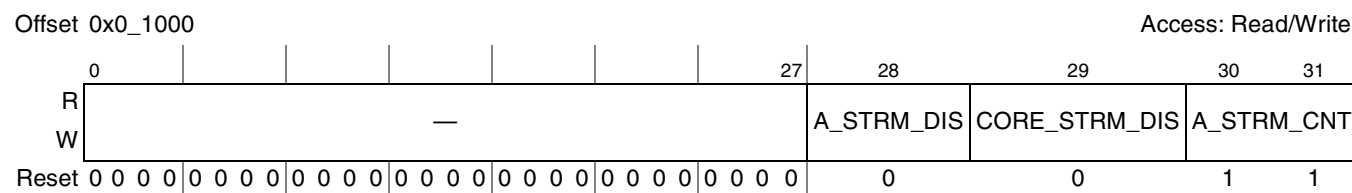
Local Memory Offset	Register	Access	Reset	Section/page
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0n00_0000	8.2.1.2/8-4
0x0_1BF8	ECM IP Block Revision Register 1	R	0x0001_0000	8.2.1.3/8-5
0x0_1BFC	ECM IP Block Revision Register 2	R	0x0000_0000	8.2.1.4/8-5
0x0_1E00	EEDR—ECM error detect register	w1c	0x0000_0000	8.2.1.5/8-6
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.6/8-7
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.7/8-7
0x0_1E10	EELADR—ECM error low address capture register	R	0x0000_0000	8.2.1.8/8-8
0x0_1E14	EEHADR—ECM error high address capture register	R	0x0000_0000	8.2.1.9/8-9

### 8.2.1 Register Descriptions

This section consists of detailed descriptions of those registers summarized in Table 8-1. Note that these registers are shown in big-endian format.

#### 8.2.1.1 ECM CCB Address Configuration Register (EEBACR)

The ECM CCB address configuration register, shown in Figure 8-2, controls arbitration and streaming policies for the CCB.



**Figure 8-2. ECM CCB Address Configuration Register (EEBACR)**

Table 8-2 describes the EEBACR fields.

**Table 8-2. EEBACR Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28	A_STRM_DIS	Controls whether the ECM allows any streaming to occur. 0 Streaming is enabled. 1 Streaming is disabled.
29	CORE_STRM_DIS	With A_STRM_DIS, controls whether the e500 core can stream commands onto the CCB. A_STRM_DIS and CORE_STRM_DIS must both be cleared for the e500 core to be enabled to stream address tenures that it masters. 0 Stream address tenures initiated by the e500 core, provided A_STRM_DIS is cleared. 1 Streaming of address tenures initiated by the e500 core not allowed.
30–31	A_STRM_CNT	Stream count. Specifies the maximum number of transactions that any master can stream (issue sequentially without preemption) on the CCB following an initial transaction. 00 Reserved 01 One transaction can be streamed with the initial transaction. 10 Two transactions can be streamed with the initial transaction. 11 Three transactions can be streamed with the initial transaction. Default.

### 8.2.1.2 ECM CCB Port Configuration Register (EEBPCR)

The ECM CCB port configuration register (EEBPCR) is shown in Figure 8-3.

Offset 0x0\_1010 Access: Read/Write

	0		6	7	8		28	29	30	31	
R	—				CPU_EN	—			CPU_RD_HI_DIS	CPU_PRI	
W											
Reset	0	0	0	0	0	0	0	0	0	0	0

*n*

**Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)**

Table 8-3 describes EEBPCR fields.

**Table 8-3. EEBPCR Field Descriptions**

Bits	Name	Description
0–6	—	Reserved
7	CPU_EN	CPU port enable. Controls boot holdoff mode when the device is an agent of an external host. Specifies whether the e500 core (CPU) port is enabled to run transactions on the CCB. The CPU boot configuration power-on reset pin (cfg_cpu_boot) determines the initial value of this bit. If the pin is sampled as a logic 1 at the negation of reset, the CPU is enabled to boot at the end of the POR sequence. Otherwise, the CPU cannot fetch its boot vector until an external host sets the CPU_EN bit. 0 Boot holdoff mode. CPU arbitration is disabled on the CCB and no bus grants are issued. 1 CPU is enabled and receives bus grants in response to bus requests for the boot vector. After this bit is set, it should not be cleared by software. It is not intended to dynamically enable and disable CPU operation. It is only intended to end boot holdoff mode. See Section 4.4.3.7, “CPU Boot Configuration,” for more information.
8–28	—	Reserved

Table 8-3. EEBPCR Field Descriptions (continued)

Bits	Name	Description
29	CPU_RD_HI_DIS	Identifies which read queue of DDR targets is assigned to the e500 core (CPU) port's read transactions (in understressed system). 0 Read high queue (higher bandwidth DDR queue) is assigned for the e500 core's read transactions 1 Read low queue (lower bandwidth DDR queue) is assigned for the e500 core's read transactions
30–31	CPU_PRI	Specifies the priority level of the e500 core 0 (CPU) port. This priority level is used to determine whether a particular port's bus request can cause the CCB arbiter to terminate another port's streaming of address tenures. 00 Lowest priority level 01 Second lowest priority level 10 Highest priority level 11 Reserved

### 8.2.1.3 ECM IP Block Revision Register 1 (EIPBRR1)

The ECM IP block revision register 1 is shown in [Figure 8-4](#).

Offset 0x0\_1BF8

Access: Read only

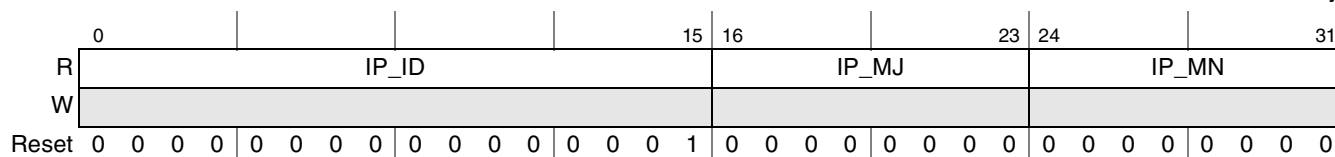


Figure 8-4. ECM IP Block Revision Register 1 (EIPBRR1)

[Table 8-4](#) describes EIPBRR1 fields.

Table 8-4. EIPBRR1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID
16–23	IP_MJ	Major revision
24–31	IP_MN	Minor revision

### 8.2.1.4 ECM IP Block Revision Register 2 (EIPBRR2)

The ECM IP block revision register 2 is shown in [Figure 8-5](#).

Offset 0x0\_1BFC

Access: Read only

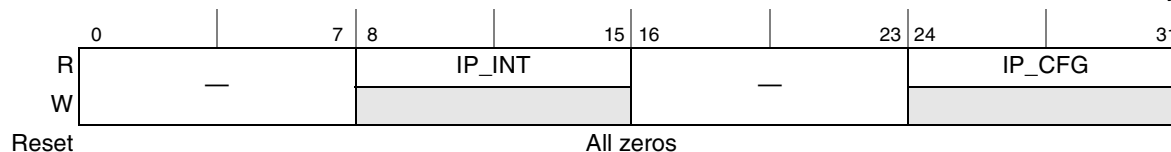


Figure 8-5. ECM IP Block Revision Register 2 (EIPBRR2)

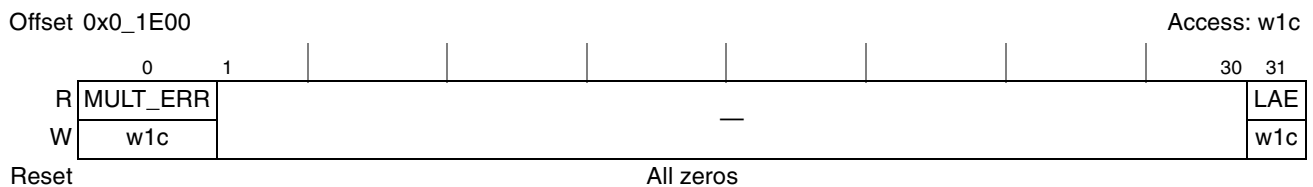
Table 8-5 describes EIPBRR2 fields.

**Table 8-5. EIPBRR2 Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

### 8.2.1.5 ECM Error Detect Register (EEDR)

The ECM error detect register (EEDR) is shown in Figure 8-6.



**Figure 8-6. ECM Error Detect Register (EEDR)**

Table 8-6 describes EEDR fields.

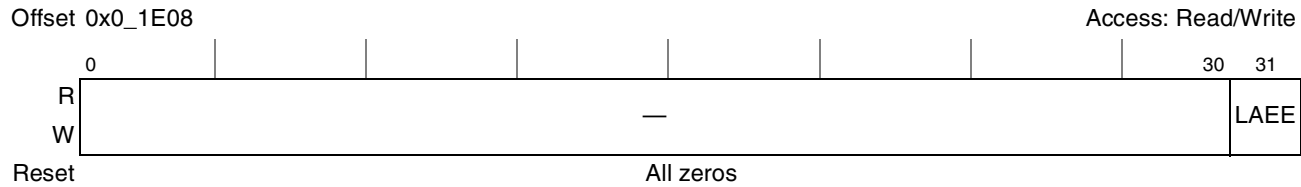
**Table 8-6. EEDR Field Descriptions**

Bits	Name	Description
0	MULT_ERR	Multiple error. Indicates the occurrence of multiple errors of the same type. Write 1 to clear. 0 Multiple errors of the same type were not detected. 1 Multiple errors of the same type were detected.
1–30	—	Reserved
31	LAE	Local access error. Write 1 to clear. Two cases can generate LAEs: <ul style="list-style-type: none"> <li>Transaction does not map to any target. In this case the ECM injects read responses (with the corrupt attribute set) and write data is dropped. Note that a read that attempts to access an unmapped target causes the assertion of <i>core_fault_in</i>, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, EEER[LAE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”</li> <li>Source and target IDs indicate that an OCN port initiated a transaction that targets an OCN port. This loopback behavior can result from programming errors where inbound ATMU window targets are inconsistent with targets configured in the local access windows for a given address range. For this type of LAE, the dispatch (to OCN target in this case) is not screened off; the LAE error is reported, but the transaction is still sent to its OCN target.</li> </ul> 0 Local access error has not occurred. 1 Local access error occurred.



### 8.2.1.6 ECM Error Enable Register (EEER)

The ECM error enable register (EEER) shown in [Figure 8-7](#) enables the reporting of error conditions to the e500 core through the internal  $\overline{int}$  interrupt signal.



**Figure 8-7. ECM Error Enable Register (EEER)**

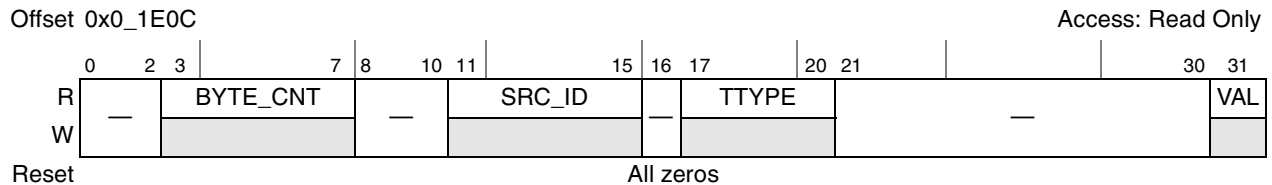
[Table 8-7](#) describes EEER fields.

**Table 8-7. EEER Field Descriptions**

Bits	Name	Description
0–30	—	Reserved
31	LAEE	Local access error enable. Note that a read that attempts to access an unmapped target causes the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If HID1[RFXE] is zero and this error occurs, LAEE must be set to ensure that an interrupt is generated. For more information, see <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1)”</a> 0 Disable reporting local access errors as interrupts. 1 Enable reporting local access errors as interrupts.

### 8.2.1.7 ECM Error Attributes Capture Register (EEATR)

The ECM error attributes capture register (EEATR) is shown in [Figure 8-8](#).



**Figure 8-8. ECM Error Attributes Capture Register (EEATR)**

[Table 8-8](#) describes EEATR fields.

**Table 8-8. EEATR Field Descriptions**

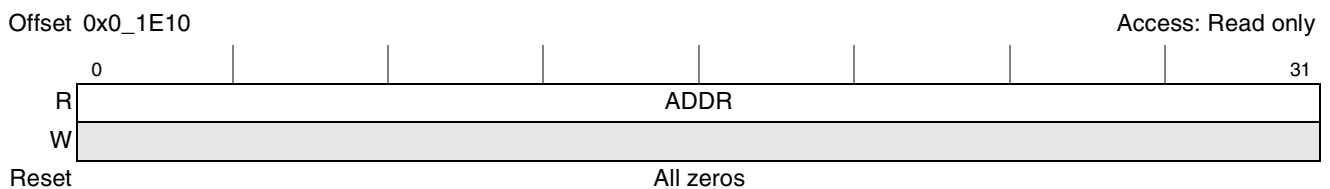
Bits	Name	Description
0–2	—	Reserved
3–7	BYTE_CNT	Byte count. Specifies the transaction byte count. 00000 32 bytes <span style="float:right">00100 4 bytes</span> 00001 1 byte <span style="float:right">01000 8 bytes</span> 00010 2 bytes <span style="float:right">10000 16 bytes</span>
8–10	—	Reserved

**Table 8-8. EEATR Field Descriptions (continued)**

Bits	Name	Description
11–15	SRC_ID	Source ID. Specifies the source device mastering the transaction. 00000 PCI interface   10100 Reserved 00001 PCI Express 2   10101 DMA 00010 PCI Express 1   10110 Reserved 00011 PCI Express 3   10111 SAP 00100–01001 Reserved   11000 eTSEC1 01010 Boot sequencer   11001 Reserved 01011 Reserved   11010 eTSEC3 01100 Reserved   11011 Reserved 01101–01111 Reserved   11100 Reserved 10000 Processor (instruction)   11101 Reserved 10001 Processor (data)   11110 Reserved 10010–10011 Reserved   11111 Reserved
16	—	Reserved
17–20	TTYPER	Transaction type. Defined as follows: 0000 Write   1001 Read with unlock 0001 Reserved   101x Reserved 0010 Write with allocate   1100 Read with clear atomic 0011 Write with allocate with lock   1101 Read with set atomic 0100 Address only transaction   1110 Read with decrement atomic 0101–0111 Reserved   1111 Read with increment atomic 1000 Read
21–30	—	Reserved
31	VAL	Register data valid. 0 ECM error attribute capture register does not contain valid information. 1 ECM error attribute capture register contains valid information.

### 8.2.1.8 ECM Error Low Address Capture Register (EELADR)

The ECM error low address capture register (EELADR) is shown in [Figure 8-9](#).



**Figure 8-9. ECM Error Low Address Capture Register (EELADR)**

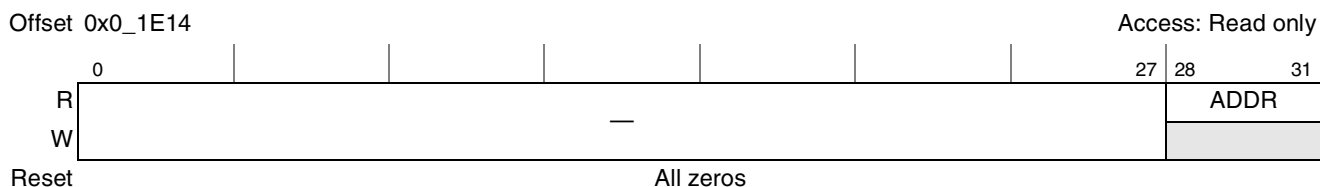
[Table 8-9](#) describes EELADR fields.

**Table 8-9. EELADR Field Descriptions**

Bits	Name	Description
0–31	ADDR	Address. Specifies the lower-order 32 bits of the 36-bit address of the transaction. Qualified by EEATR[VAL].

### 8.2.1.9 ECM Error High Address Capture Register (EEHADR)

The ECM error high address capture register (EEHADR) is shown in [Figure 8-10](#).



**Figure 8-10. ECM Error High Address Capture Register (EEHADR)**

[Table 8-10](#) describes EEHADR fields.

**Table 8-10. EEHADR Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Address. Specifies the high-order 4 bits of the 36-bit address of the transaction. Qualified by EEATR[VAL].

## 8.3 Functional Description

The following is a very general discussion of ECM operation.

### 8.3.1 I/O Arbiter

[Figure 8-1](#) shows the I/O arbiter block that manages I/O-initiated address tenure requests arriving on the request buses. Four request buses compete for access to the ECM, which can only process one request at a time. The ECM uses two factors to select the winning request bus: the primary factor is requested bandwidth and the secondary factor is longest waiting/least recently granted status. By default all requesters start requesting low levels of bandwidth. A starvation avoidance algorithm ensures that low bandwidth requesters make forward progress in the presence of high bandwidth requesters. The transaction from the winning request bus competes with e500 core requests for the CCB and entry into the transaction queue.

### 8.3.2 CCB Arbiter

[Figure 8-1](#) shows the CCB arbiter block coordinating the entry of new transactions into the ECM's transaction queue. It handles arbitration for requests to use the CCB from the e500 core and the winning request bus and consequently controls when these new transactions can enter the transaction queue.

Because the CCB bus operates most efficiently when it streams commands from one initiator, the CCB arbiter alternates grants between streams of transactions from the e500 core and from the winner of the I/O arbiter. The length of a stream (number of back-to-back transactions) is limited by the A\_STRM\_CNT field in the EEBAACR register. However, the arbiter also uses the priority of the requests to limit streaming. If the priority of a new request is higher than that of a stream in progress, then the higher priority transaction will interrupt the other stream. The priority of e500 transactions is set by the CPU\_PRI field

in EEBPCR register. Depending how the CPU\_RD\_HI\_DIS field in EEBPCR register is set, read transactions from the e500 core are initially assigned to either the higher- or lower-bandwidth queue of the DDR target.

### 8.3.3 Transaction Queue

The ECM's transaction queue performs four basic functions: arbitration across the e500 core and I/O masters, target mapping and dispatching, enforcement of ordering, and enforcement of coherency. The address of each transaction is compared against each local access window, and the transaction is then routed to the appropriate target interface associated with the local access window that the address hits within. Even though the CCB and ECM allow the pipelining of transactions, the address tenures of all transactions issued from I/O masters (masters other than the e500 core) may still be ordered. For those transactions accessing address space marked as snoopable, or space that may be cached by the e500 core, the ECM enforces coherency, snooping those transactions on the CCB, and taking castouts from the e500 core as is necessary.

### 8.3.4 Global Data Multiplexor

Figure 8-1 shows how the global data multiplexor takes data bus connections and multiplexes them onto one 128-bit global data bus. The global data mux allows initiators of write transactions to route data to their targets and read targets to return data to the initiators.

### 8.3.5 CCB Interface

Figure 8-1 shows the CCB interface for both CCB address and data tenures. This interface formats CCB address tenures for the ECM transaction queue. It also contains the queueing and buffering needed to manage outstanding CCB data tenures. The buffers receive e500 core-initiated write and I/O-initiated read data (that hit in the L2/SRAM module) from the e500 write (128-bit wide) and read (128-bit wide) data buses and route them through the global data mux to the global data bus. The buffers also receive e500 core-initiated read and I/O-initiated write data (that hit in the L2/SRAM module) from the global data bus and forward them onto the CCB data bus (64 bits).

## 8.4 Initialization/Application Information

If the e500 core is used to initialize the MPC853E, the CPU boot configuration power-on reset pin should be pulled high to initially set EEBPCR[CPU\_EN]. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on power-up reset initialization.

If any device other than the e500 core (such as PCI Express) is used to initialize the MPC853E, the CPU boot configuration power-on reset pin should be pulled low to initially clear EEBPCR[CPU\_EN]. This prevents the e500 core from accessing any configuration registers or local memory space during initialization. However, in any such system, one step near the end of the initialization routine must set EEBPCR[CPU\_EN] to re-enable the e500 core. Note that for basic functionality, EEBPCR[CPU\_EN] is the only field that must be written (provided a device other than the e500 core is used to initialize the device) in the ECM.

EEBPCR[CPU\_PRI] specifies the priority level associated with all e500 core initiated transactions. This value allows users running time-critical applications to adjust the average response latency of transactions initiated by the core compared to those initiated by I/O masters. This priority level affects whether e500 core requests can interrupt the streaming of address tenures initiated by (the ECM on behalf of) I/O masters. Only transactions with a priority greater than the current CCB transaction can interrupt streaming. The higher the core's priority, the lower the average latency needed for it to obtain bus grants from the ECM, because it can interrupt lower priority streaming. The default value of zero gives all core-initiated transactions the lowest priority, which prevents the core from interrupting I/O master transaction streams.

EEBACR[A\_STRM\_CNT] allows users to balance response latency with throughput and should prove useful in tuning systems with multiple time-critical tasks. The default value of 0b11 causes the ECM to attempt to stream as many as four transactions initiated from the same CCB master. Increasing this value increases the maximum number of transactions that may be streamed together from any one CCB master. Raising this value can increase throughput for high priority transactions, but may increase latency for lower priority transactions from another CCB master. Note that the e500 core must also have streaming enabled (through HID1[ASTME]) for the CCB to stream.



# Chapter 9

## DDR Memory Controller

### 9.1 Introduction

The fully programmable DDR SDRAM controller supports most JEDEC standard x8, x16, or x32 DDR2 and DDR memories available. In addition, unbuffered and registered DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

#### NOTE

In this chapter, the word ‘bank’ refers to a physical bank specified by a chip select; ‘logical bank’ refers to one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the 2 or 3 bits on the bank address (MBA) pins during a memory access.

[Figure 9-1](#) is a high-level block diagram of the DDR memory controller with its associated interfaces. [Section 9.5, “Functional Description,”](#) contains detailed figures of the controller.

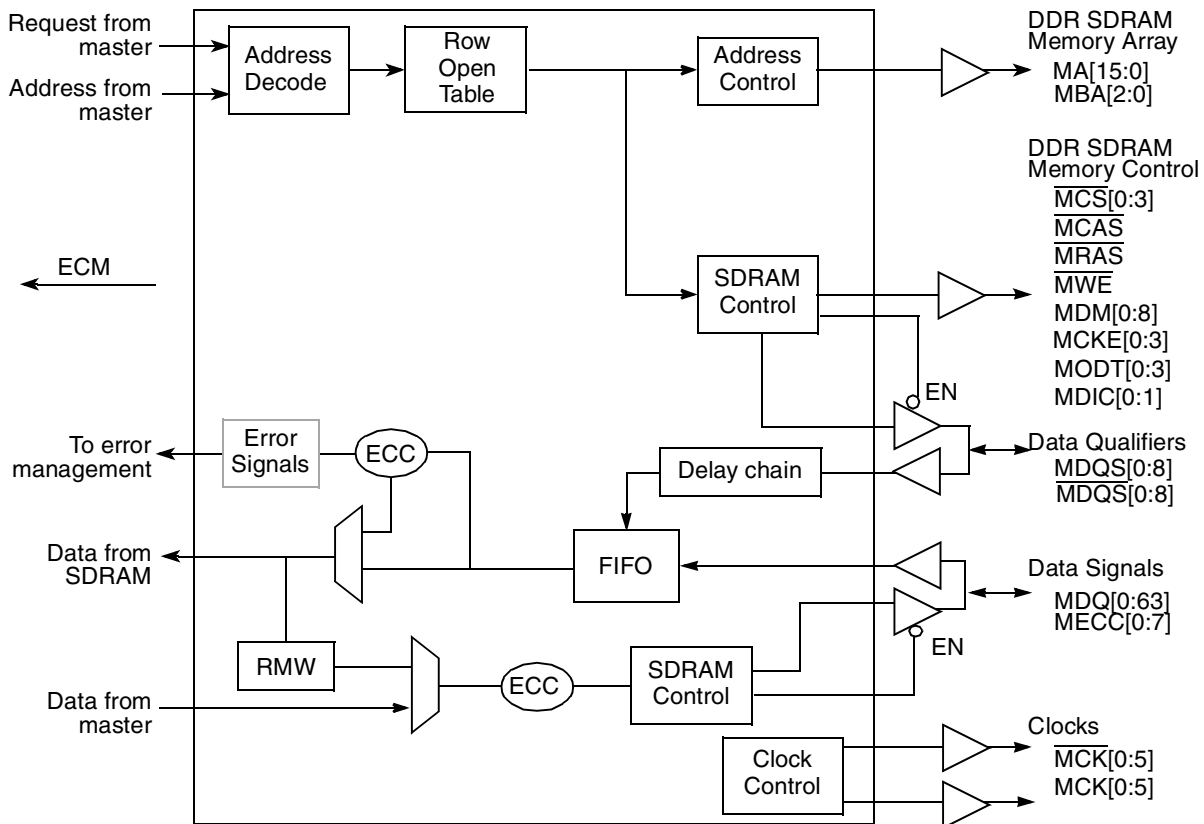


Figure 9-1. DDR Memory Controller Simplified Block Diagram

## 9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR SDRAM
- 64-/72-bit SDRAM data bus. 32-/40-bit SDRAM for DDR and DDR2
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - As many as four physical banks (chip selects), each bank independently addressable
  - 64-Mbit to 4-Gbit devices depending on internal device configuration with x8/x16/x32 data ports (no direct x4 support)
  - Unbuffered and registered DIMMs
- Chip select interleaving support
- Support for data mask signals and read-modify-write for sub-double-word writes. Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence



- Automatic DRAM data initialization
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

## 9.2.1 Modes of Operation

The DDR memory controller supports the following modes:

- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing DDR\_SDRAM\_INTERVAL[BSTOPRE] causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting CS<sub>n</sub>\_CONFIG[AP\_n\_EN].

## 9.3 External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

### 9.3.1 Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 9-1 shows how DDR memory controller external signals are grouped. The device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

**Table 9-1. DDR Memory Interface Signal Summary**

Name	Function/Description	Reset	Pins	I/O
MDQ[0:63]	Data bus	All zeros	64	I/O
MDQS[0:8]	Data strobes	All zeros	9	I/O
$\overline{\text{MDQS}}[0:8]$	Complement data strobes	All ones	9	I/O
MECC[0:7]	Error checking and correcting	All zeros	8	I/O
$\overline{\text{MCAS}}$	Column address strobe	One	1	O
MA[15:0]	Address bus	All zeros	16	O
MBA[2:0]	Logical bank address	All zeros	3	O

**Table 9-1. DDR Memory Interface Signal Summary (continued)**

Name	Function/Description	Reset	Pins	I/O
$\overline{\text{MCS}}[0:3]$	Chip selects	All ones	4	O
$\overline{\text{MWE}}$	Write enable	One	1	O
$\overline{\text{MRAS}}$	Row address strobe	One	1	O
MDM[0:8]	Data mask	All zeros	9	O
MCK[0:5]	DRAM clock outputs	All zeros	6	O
$\overline{\text{MCK}}[0:5]$	DRAM clock outputs (complement)	All zeros	6	O
MCKE[0:3]	DRAM clock enable	All zeros	4	O
MODT[0:3]	DRAM on-die termination	All zeros	4	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	O
MDIC[0:1]	Driver impedance calibration	b10	2	I/O

Table 9-2 shows the memory address signal mappings.

**Table 9-2. Memory Address Signal Mappings**

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA15	A15
	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR) <sup>1</sup>
	MA9	A9
	MA8	A8 (alternate AP for DDR) <sup>2</sup>
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1

Table 9-2. Memory Address Signal Mappings (continued)

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

<sup>1</sup> Auto-precharge for DDR signaled on A10 when DDR\_SDRAM\_CFG[PCHB8] = 0

<sup>2</sup> Auto-precharge for DDR signaled on A8 when DDR\_SDRAM\_CFG[PCHB8] = 1

## 9.3.2 Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

### 9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O		As outputs for the bidirectional data bus, these signals operate as described below.
		<b>State Meaning</b>	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		<b>Timing</b>	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I		As inputs for the bidirectional data bus, these signals operate as described below.
		<b>State Meaning</b>	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
		<b>Timing</b>	Assertion/Negation—The DDR SDRAM drives data during a READ transaction. High impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.

**Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description	
MDQS[0:8]/ MDQS[0:8]	I/O	Data strobes. Inputs with read data, outputs with write data. The data strobes may be single ended or differential.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		<b>State Meaning</b>	Asserted/Negated—Driven high when positive capture data is transmitted and driven low when negative capture data is transmitted. Centered in the data “eye” for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See <a href="#">Table 9-38</a> for byte lane assignments.
		<b>Timing</b>	Assertion/Negation—If a WRITE command is registered at clock edge $n$ , data strobes at the DRAM assert centered in the data eye on clock edge $n + 1$ . See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		<b>State Meaning</b>	Asserted/Negated—Driven high when positive capture data is received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See <a href="#">Table 9-38</a> for byte lane assignments.
<b>Timing</b>		Assertion/Negation—If a READ command is registered at clock edge $n$ , and the latency is programmed in <code>TIMING_CFG_1[CASLAT]</code> to be $m$ clocks, data strobes at the DRAM assert coincident with the data on clock edge $n + m$ . See the JEDEC DDR SDRAM specification for more information.	
MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller’s bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See <a href="#">Section 9.5.11, “Error Checking and Correcting (ECC),”</a> and <a href="#">Section 21.4.2.2, “Debug Information on ECC Pins,”</a> for more details.	
		<b>State Meaning</b>	Asserted/Negated—Represents the state of ECC being driven by the DDR controller on writes.
		<b>Timing</b>	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		<b>State Meaning</b>	Asserted/Negated—Represents the state of ECC being driven by the DDR SDRAMs on reads.
<b>Timing</b>		Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ	

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MA[15:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[15:0] carry 16 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		<b>State Meaning</b>	Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See <a href="#">Table 9-41</a> <a href="#">Table 9-42</a> for a complete description of the mapping of these signals.
		<b>Timing</b>	Assertion/Negation—The address is always driven when the memory controller is enabled. It is valid when a transaction is driven to DRAM (when $\overline{MCSn}$ is active). High impedance—When the memory controller is disabled
MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		<b>State Meaning</b>	Asserted/Negated—Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. <a href="#">Table 9-41</a> <a href="#">Table 9-42</a> describes the mapping of these signals in all cases.
		<b>Timing</b>	Assertion/Negation—Same timing as $MA_n$ High impedance—Same timing as $MA_n$
$\overline{MCAS}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{MCAS}$ is asserted for read or write transactions and for mode register set, refresh, and precharge commands.	
		<b>State Meaning</b>	Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See <a href="#">Table 9-46</a> for more information on the states required on $\overline{MCAS}$ for various other SDRAM commands. Negated—The column address is not guaranteed to be valid.
		<b>Timing</b>	Assertion/Negation—Assertion and negation timing is directed by the values described in <a href="#">Section 9.4.1.4</a> , “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” <a href="#">Section 9.4.1.5</a> , “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” <a href="#">Section 9.4.1.6</a> , “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and <a href="#">Section 9.4.1.3</a> , “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).” High impedance— $\overline{MCAS}$ is always driven unless the memory controller is disabled.
$\overline{MRAS}$	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.	
		<b>State Meaning</b>	Asserted—Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See <a href="#">Table 9-46</a> for more information on the states required on $\overline{MRAS}$ for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		<b>Timing</b>	Assertion/Negation—Assertion and negation timing is directed by the values described in <a href="#">Section 9.4.1.4</a> , “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” <a href="#">Section 9.4.1.5</a> , “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” <a href="#">Section 9.4.1.6</a> , “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and <a href="#">Section 9.4.1.3</a> , “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).” High impedance— $\overline{MRAS}$ is always driven unless the memory controller is disabled.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{MCS}}[0:3]$	O	Chip selects. Four chip selects supported by the memory controller.	
		<b>State Meaning</b>	Asserted—Selects a physical SDRAM bank to perform a memory operation as described in <a href="#">Section 9.4.1.1</a> , “Chip Select Memory Bounds ( $\text{CS}_n\text{_BNDS}$ ),” and <a href="#">Section 9.4.1.2</a> , “Chip Select Configuration ( $\text{CS}_n\text{_CONFIG}$ ).” The DDR controller asserts one of the $\overline{\text{MCS}}[0:3]$ signals to begin a memory cycle. Negated—Indicates no SDRAM action during the current cycle.
		<b>Timing</b>	Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in $\text{TIMING\_CFG}_0\text{--TIMING\_CFG}_3$ . High impedance—Always driven unless the memory controller is disabled.
$\overline{\text{MWE}}$	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.	
		<b>State Meaning</b>	Asserted—Indicates a memory write operation. See <a href="#">Table 9-46</a> for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands. Negated—Indicates a memory read operation.
		<b>Timing</b>	Assertion/Negation—Similar timing as $\overline{\text{MRA}}\text{S}$ and $\overline{\text{MCAS}}$ . Used for write commands. High impedance— $\overline{\text{MWE}}$ is always driven unless the memory controller is disabled.
$\text{MDM}[0:8]$	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. $\text{MDM}_0$ corresponds to the most significant byte (MSB) and $\text{MDM}_7$ corresponds to the LSB, while $\text{MDM}_8$ corresponds to the ECC byte. <a href="#">Table 9-38</a> shows byte lane encodings.	
		<b>State Meaning</b>	Asserted—Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the $\text{MDM}_n$ signals are active-high for the DDR controller. $\text{MDM}_n$ is part of the DDR command encoding. Negated—Allows the corresponding byte to be read from or written to the SDRAM.
		<b>Timing</b>	Assertion/Negation—Same timing as $\text{MDQx}$ as outputs. High impedance—Always driven unless the memory controller is disabled.
$\text{MODT}[0:3]$	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. $\text{MODT}[0:3]$ represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		<b>State Meaning</b>	Asserted/Negated—Represents the ODT driven by the DDR memory controller.
		<b>Timing</b>	Assertion/Negation—Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the $\text{CS}_n\text{_CONFIG}[\text{ODT\_RD\_CFG}]$ and $\text{CS}_n\text{_CONFIG}[\text{ODT\_WR\_CFG}]$ fields. High impedance—Always driven.
$\text{MDIC}[0:1]$	I/O	Driver impedance calibration. Note that the MDIC signals require the use of 18.2- $\Omega$ precision 1% resistors; $\text{MDIC}_0$ must be pulled to GND, while $\text{MDIC}_1$ must be pulled to $\text{GV}_{\text{DD}}$ . See <a href="#">Section 19.4.1.21</a> , “DDR Control Driver Register ( $\text{DDRCDR}$ ),” for more information on these signals.	
		<b>State Meaning</b>	These pins are used for automatic calibration of the DDR IOs.
		<b>Timing</b>	These will be driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.

### 9.3.2.2 Clock Interface Signals

Table 9-4 contains the detailed descriptions of the clock signals of the DDR controller.

**Table 9-4. Clock Signals—Detailed Signal Descriptions**

Signal	I/O	Description
MCK[0:5], $\overline{\text{MCK}}[0:5]$	O	DRAM clock outputs and their complements. See <a href="#">Section 9.5.4.1, “Clock Distribution.”</a>
		<b>State Meaning</b> Asserted/Negated—The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		<b>Timing</b> Assertion/Negation—Timing is controlled by the DDR_CLK_CNTL register at offset 0x130.
MCKE[0:3]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0:3] can be negated to stop clocking the DDR SDRAM.
		<b>State Meaning</b> Asserted—Clocking to the SDRAM is enabled. Negated—Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or $\overline{\text{MCK}}$ . MCK/ $\overline{\text{MCK}}$ are don't cares while MCKE[0:3] are negated.
		<b>Timing</b> Assertion/Negation—Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Will be asserted again when exiting dynamic power management or self refresh. High impedance—Always driven.

### 9.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in [Section 21.4.2, “DDR SDRAM Interface Debug.”](#)

## 9.4 Memory Map/Register Definition

Table 9-5 shows the register memory map for the DDR memory controller.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 9-5. DDR Memory Controller Memory Map**

Offset	Register	Access	Reset	Section/Page
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	<a href="#">9.4.1.1/9-11</a>
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	<a href="#">9.4.1.1/9-11</a>
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	<a href="#">9.4.1.1/9-11</a>
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	<a href="#">9.4.1.1/9-11</a>



Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-11
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	9.4.1.2/9-11
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	9.4.1.2/9-11
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	9.4.1.2/9-11
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-13
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-14
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-16
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-18
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-20
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-23
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-25
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-26
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-26
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-29
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-29
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x140– 0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-30
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	9.4.1.16/9-31
0x150– 0xBF4	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0xn <sub>nnnn</sub> _n <sub>nnnn</sub> <sup>1</sup>	9.4.1.17/9-32
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00n <sub>n</sub> _00n <sub>n</sub> <sup>1</sup>	9.4.1.18/9-32
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.19/9-33
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.20/9-33
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.21/9-34
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.22/9-34
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.23/9-35
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.24/9-35
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.25/9-35
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.26/9-36
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.27/9-37
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.28/9-38
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.29/9-39
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	9.4.1.30/9-40
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.31/9-40

<sup>1</sup> Implementation-dependent reset values are listed in specified section/page.



## 9.4.1 Register Descriptions

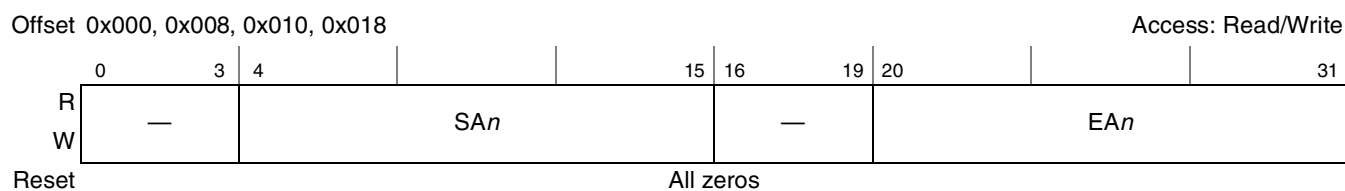
This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

### 9.4.1.1 Chip Select Memory Bounds ( $CS_n\_BNDS$ )

The chip select bounds registers ( $CS_n\_BNDS$ ) define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in  $CS_n\_BNDS$  should equal the size of physical DRAM. Also, note that  $EAn$  must be greater than or equal to  $SAn$ .

If chip select interleaving is enabled, all fields in the lower interleaved chip select will be used, and the other chip selects' bounds registers will be unused. For example, if chip selects 0 and 1 are interleaved, all fields in  $CS0\_BNDS$  will be used, and all fields in  $CS1\_BNDS$  will be unused.

$CS_n\_BNDS$  are shown in [Figure 9-2](#).



**Figure 9-2. Chip Select Bounds Registers ( $CS_n\_BNDS$ )**

[Table 9-6](#) describes the  $CS_n\_BNDS$  register fields.

**Table 9-6.  $CS_n\_BNDS$  Field Descriptions**

Bits	Name	Description
0–3	—	Reserved
4–15	$SAn$	Starting address for chip select (bank) $n$ . This value is compared against the 12 msbs of the 36-bit address.
16–19	—	Reserved
20–31	$EAn$	Ending address for chip select (bank) $n$ . This value is compared against the 12 msbs of the 36-bit address.

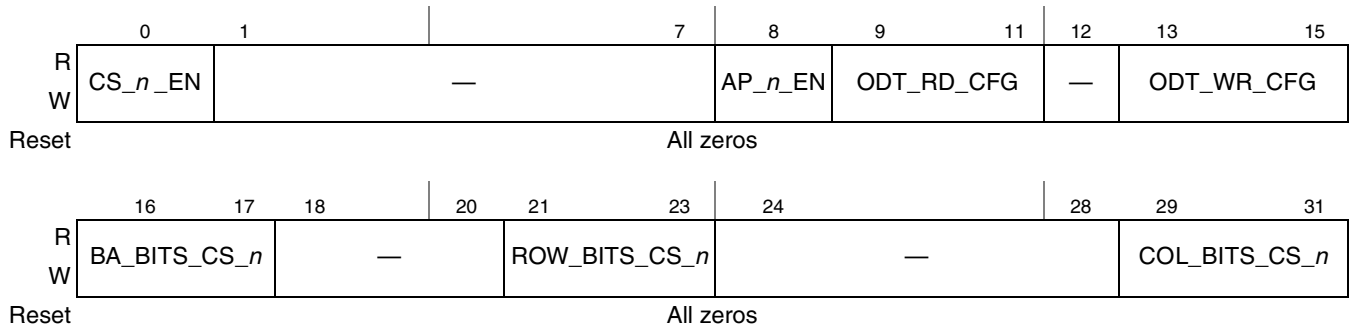
### 9.4.1.2 Chip Select Configuration ( $CS_n\_CONFIG$ )

The chip select configuration ( $CS_n\_CONFIG$ ) registers shown in [Figure 9-3](#) enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because  $CS_n\_CONFIG[ROW\_BITS\_CS\_n, COL\_BITS\_CS\_n]$  establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused, with the exception of the  $ODT\_RD\_CFG$  and  $ODT\_WR\_CFG$  fields. For example, if chip selects 0 and 1 are interleaved, all fields in  $CS0\_CONFIG$  will be used, but only the  $ODT\_RD\_CFG$  and  $ODT\_WR\_CFG$  fields in  $CS1\_CONFIG$  will be used.

Offset 0x080, 0x084, 0x088, 0x08C

Access: Read/Write



**Figure 9-3. Chip Select Configuration Register (CS<sub>n</sub>\_CONFIG)**

Table 9-7 describes the CS<sub>n</sub>\_CONFIG register fields.

**Table 9-7. CS<sub>n</sub>\_CONFIG Field Descriptions**

Bits	Name	Description
0	CS <sub>n</sub> _EN	Chip select <i>n</i> enable 0 Chip select <i>n</i> is not active 1 Chip select <i>n</i> is active and assumes the state set in CS <sub>n</sub> _BNDS.
1–7	—	Reserved
8	AP <sub>n</sub> _EN	Chip select <i>n</i> auto-precharge enable 0 Chip select <i>n</i> will only be auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select <i>n</i> will always issue an auto-precharge for read and write transactions.
9–11	ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS <sub>n</sub> 010 Assert ODT only during reads to other chip selects  011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads 101–111 Reserved
12	—	Reserved
13–15	ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT_WR_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS <sub>n</sub> 010 Assert ODT only during writes to other chip selects 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes 101–111 Reserved
16–17	BA_BITS_CS <sub>n</sub>	Number of bank bits for SDRAM on chip select <i>n</i> . These bits correspond to the sub-bank bits driven on MBA <sub>n</sub> in Table 9-42, Table 9-41, and Table 9-42. 00 2 logical bank bits 01 3 logical bank bits 10–11 Reserved

**Table 9-7. CS<sub>n</sub>\_CONFIG Field Descriptions (continued)**

Bits	Name	Description
18–20	—	Reserved
21–23	ROW_BITS_CS_ <i>n</i>	Number of row bits for SDRAM on chip select <i>n</i> . See Table 9-42 Table 9-41 and Table 9-42 for details. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits 101–111 Reserved
24–28	—	Reserved
29–31	COL_BITS_CS_ <i>n</i>	Number of column bits for SDRAM on chip select <i>n</i> . For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100–111 Reserved

### 9.4.1.3 DDR SDRAM Timing Configuration 3 (TIMING\_CFG\_3)

DDR SDRAM timing configuration register 3, shown in Figure 9-4, sets the extended refresh recovery time, which is combined with TIMING\_CFG\_1[REFREC] to determine the full refresh recovery time.

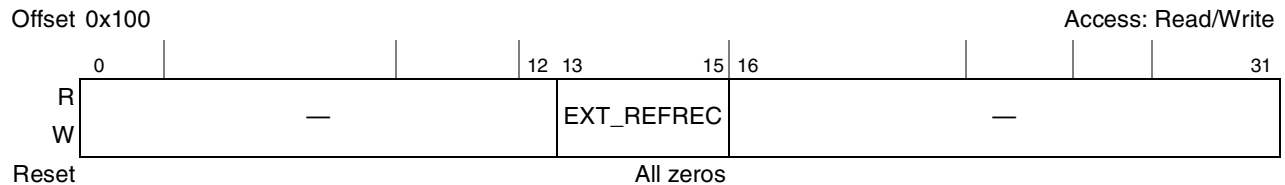
**Figure 9-4. DDR SDRAM Timing Configuration 3 (TIMING\_CFG\_3)**

Table 9-8 describes TIMING\_CFG\_3 fields.

**Table 9-8. TIMING\_CFG\_3 Field Descriptions**

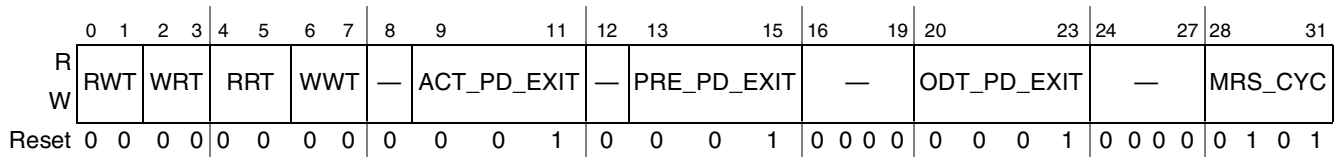
Bits	Name	Description
0–12	—	Reserved, should be cleared.
13–15	EXT_REFREC	Extended refresh recovery time ( $t_{RFC}$ ). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain a 7bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7bit value of the refresh recovery. $t_{RFC} = \{EXT\_REFREC \parallel REFREC\} + 8$ , such that $t_{RFC}$ is calculated as follows: 000 0 clocks 001 16 clocks 010 32 clocks 011 48 clocks 100 64 clocks 101 80 clocks 110 96 clocks 111 112 clocks
16–31	—	Reserved, should be cleared.

### 9.4.1.4 DDR SDRAM Timing Configuration 0 (TIMING\_CFG\_0)

DDR SDRAM timing configuration register 0, shown in Figure 9-5, sets the number of clock cycles between various SDRAM control commands.

Offset 0x104

Access: Read/Write



**Figure 9-5. DDR SDRAM Timing Configuration 0 (TIMING\_CFG\_0)**

Table 9-9 describes TIMING\_CFG\_0 fields.

**Table 9-9. TIMING\_CFG\_0 Field Descriptions**

Bits	Name	Description
0–1	RWT	Read-to-write turnaround ( $t_{RTW}$ ). Specifies how many extra cycles will be added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller will determine the read-to-write turnaround as $CL - WL + BL/2 + 2$ . In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks

Table 9-9. TIMING\_CFG\_0 Field Descriptions (continued)

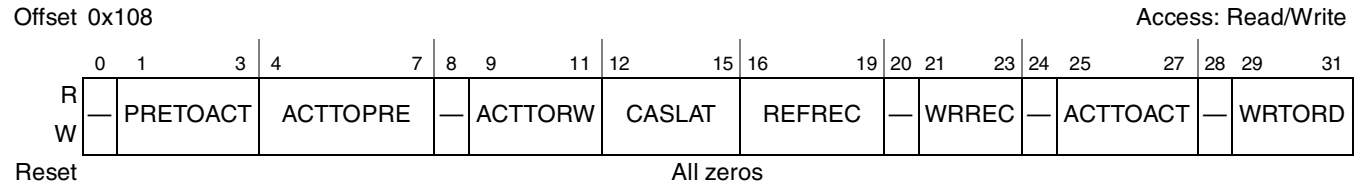
Bits	Name	Description
2–3	WRT	Write-to-read turnaround. Specifies how many extra cycles will be added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller will determine the write-to-read turnaround as $WL - CL + BL/2 + 1$ . In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles will be added between reads to different chip selects. As a default, 3 cycles will be required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles will be the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles will be added between writes to different chip selects. As a default, 2 cycles will be required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles will be the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
8	—	Reserved, should be cleared.
9–11	ACT_PD_EXIT	Active powerdown exit timing ( $t_{XARD}$ and $t_{XARDS}$ ). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12	—	Reserved, should be cleared.
13–15	PRE_PD_EXIT	Precharge powerdown exit timing ( $t_{XP}$ ). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
16–19	—	Reserved, should be cleared.

**Table 9-9. TIMING\_CFG\_0 Field Descriptions (continued)**

Bits	Name	Description
20–23	ODT_PD_EXIT	ODT powerdown exit timing ( $t_{AXPD}$ ). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. 0000 0 clock   1000 8 clocks 0001 1 clock   1001 9 clocks 0010 2 clocks   1010 10 clocks 0011 3 clocks   1011 11 clocks 0100 4 clocks   1100 12 clocks 0101 5 clocks   1101 13 clocks 0110 6 clocks   1110 14 clocks 0111 7 clocks   1111 15 clocks
24–27	—	Reserved, should be cleared.
28–31	MRS_CYC	Mode register set cycle time ( $t_{MRD}$ ). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. 0000 Reserved   1000 8 clocks 0001 1 clock   1001 9 clocks 0010 2 clocks   1010 10 clocks 0011 3 clocks   1011 11 clocks 0100 4 clocks   1100 12 clocks 0101 5 clocks   1101 13 clocks 0110 6 clocks   1110 14 clocks 0111 7 clocks   1111 15 clocks

**9.4.1.5 DDR SDRAM Timing Configuration 1 (TIMING\_CFG\_1)**

DDR SDRAM timing configuration register 1, shown in Figure 9-6, sets the number of clock cycles between various SDRAM control commands.



**Figure 9-6. DDR SDRAM Timing Configuration 1 (TIMING\_CFG\_1)**

Table 9-10 describes TIMING\_CFG\_1 fields.

**Table 9-10. TIMING\_CFG\_1 Field Descriptions**

Bits	Name	Description
0	—	Reserved, should be cleared.

Table 9-10. TIMING\_CFG\_1 Field Descriptions (continued)

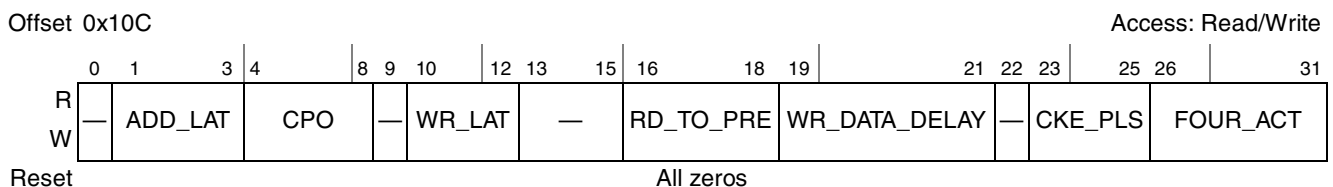
Bits	Name	Description
1–3	PRETOACT	Precharge-to-activate interval ( $t_{RP}$ ). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed.  000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
4–7	ACTTOPRE	Activate to precharge interval ( $t_{RAS}$ ). Determines the number of clock cycles from an activate command until a precharge command is allowed.  0000 16 clocks                      0101 5 clocks 0001 17 clocks                      0110 6 clocks 0010 18 clocks                      0111 7 clocks 0011 19 clocks                      ... 0100 4 clocks                        1111 15 clocks
8	—	Reserved, should be cleared.
9–11	ACTTORW	Activate to read/write interval for SDRAM ( $t_{RCD}$ ). Controls the number of clock cycles from an activate command until a read or write command is allowed.  000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12–15	CASLAT	$\overline{MCAS}$ latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge $n$ and the latency is $m$ clocks, data is available nominally coincident with clock edge $n + m$ . This value must be programmed at initialization as described in <a href="#">Section 9.4.1.8, “DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2).”</a>  0000 Reserved                      1000 4.5 clocks 0001 1 clock                        1001 5 clocks 0010 1.5 clocks                    1010 5.5 clocks 0011 2 clocks                      1011 6 clocks 0100 2.5 clocks                    1100 6.5 clocks 0101 3 clocks                      1101 7 clocks 0110 3.5 clocks                    1110 7.5 clocks 0111 4 clocks                      1111 8 clocks

**Table 9-10. TIMING\_CFG\_1 Field Descriptions (continued)**

Bits	Name	Description
16–19	REFREC	Refresh recovery time ( $t_{RFC}$ ). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7-bit value of the refresh recovery, such that $t_{RFC}$ is calculated as follows: $t_{RFC} = \{EXT\_REFREC \parallel REFREC\} + 8$ .  0000 8 clocks                      0011 11 clocks 0001 9 clocks                      ... 0010 10 clocks                      1111 23 clocks
20	—	Reserved, should be cleared.
21–23	WRREC	Last data to precharge minimum interval ( $t_{WR}$ ). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed.  000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
24	—	Reserved, should be cleared.
25–27	ACTTOACT	Activate-to-activate interval ( $t_{RRD}$ ). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select).  000 Reserved                      100 4 clocks 001 1 clock                      101 5 clocks 010 2 clocks                      110 6 clocks 011 3 clocks                      111 7 clocks
28	—	Reserved, should be cleared.
29–31	WRTORD	Last write data pair to read command issue interval ( $t_{WTR}$ ). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank.  000 Reserved                      100 4 clocks 001 1 clock                      101 5 clocks 010 2 clocks                      110 6 clocks 011 3 clocks                      111 7 clocks

### 9.4.1.6 DDR SDRAM Timing Configuration 2 (TIMING\_CFG\_2)

DDR SDRAM timing configuration 2, shown in [Figure 9-7](#), sets the clock delay to data for writes.



**Figure 9-7. DDR SDRAM Timing Configuration 2 Register (TIMING\_CFG\_2)**



Table 9-11 describes the TIMING\_CFG\_2 fields.

**Table 9-11. TIMING\_CFG\_2 Field Descriptions**

Bits	Name	Description
0	—	Reserved
1–3	ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. (DDR2-specific) 000 0 clocks 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 Reserved 111 Reserved
4–8	CPO <sup>1</sup>	MCAS-to-preamble override. Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. For these decodings, “READ_LAT” is equal to the CAS latency plus the additive latency. 00000 READ_LAT + 1                      01100 READ_LAT + 5/2 00001 Reserved                            01101 READ_LAT + 11/4 00010 READ_LAT                            01110 READ_LAT + 3 00011 READ_LAT + 1/4                    01111 READ_LAT + 13/4 00100 READ_LAT + 1/2                    10000 READ_LAT + 7/2 00101 READ_LAT + 3/4                    10001 READ_LAT + 15/4 00110 READ_LAT + 1                      10010 READ_LAT + 4 00111 READ_LAT + 5/4                    10011 READ_LAT + 17/4 01000 READ_LAT + 3/2                    10100 READ_LAT + 9/2 01001 READ_LAT + 7/4                    10101 READ_LAT + 19/4 01010 READ_LAT + 2                      10110–11111 Reserved 01011 READ_LAT + 9/4
9	—	Reserved
10–12	WR_LAT	Write latency. Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT; the write latency for DDR1 is 1. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
13–15	—	Reserved
16–18	RD_TO_PRE	Read to precharge ( $t_{RTP}$ ). For DDR2, with a non-zero ADD_LAT value, takes a minimum of ADD_LAT + $t_{RTP}$ cycles between read and precharge. For DDR1 with burst length of 4, must be set to 010; for DDR1 with burst length of 8, must be set to 100. 000 Reserved                                100 4 cycles 001 1 cycle                                    101–111 Reserved 010 2 cycles 011 3 cycles

**Table 9-11. TIMING\_CFG\_2 Field Descriptions (continued)**

Bits	Name	Description
19–21	WR_DATA_DELAY	Write command to write data strobe timing adjustment. Controls the amount of delay applied to the data and data strobes for writes. See <a href="#">Section 9.5.7, “DDR SDRAM Write Timing Adjustments,”</a> for details. 000 0 clock delay                                      100 1 clock delay 001 1/4 clock delay                                    101 5/4 clock delay 010 1/2 clock delay                                    110 3/2 clock delay 011 3/4 clock delay                                    111 Reserved
22	—	Reserved
23–25	CKE_PL5	Minimum CKE pulse width ( $t_{CKE}$ ). Can be set to 001 for DDR1. 000 Reserved    011 3 cycles 001 1 cycle    100 4 cycles 010 2 cycles    101–111 Reserved
26–31	FOUR_ACT	Window for four activates ( $t_{FAW}$ ). This is applied to DDR2 with eight logical banks only. Must be set to 000001 for DDR1. 000000 Reserved    ... 000001 1 cycle    010011 19 cycles 000010 2 cycles    010100 20 cycles 000011 3 cycles    010101–111111 Reserved 000100 4 cycles

<sup>1</sup> For CPO decodings other than 00000 and 11111, ‘READ\_LAT’ is rounded up to the next integer value.

### 9.4.1.7 DDR SDRAM Control Configuration (DDR\_SDRAM\_CFG)

The DDR SDRAM control configuration register, shown in [Figure 9-8](#), enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Offset 0x110

Access: Read/Write

R	0	1	2	3	4	5	7	8	9	10	11	12	13	14	15		
W	MEM_EN	SREN	ECC_EN	RD_EN	—	SDRAM_TYPE		—	DYN_PWR		—	32_BE	8_BE	NCAP	—		
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		
R	16		17			23	24		25	26		27	28	29	30		31
W	2T_EN		BA_INTLV_CTL			—	x32_EN		PCHB8	HSE	—	MEM_HALT	BI				
Reset	All zeros																

**Figure 9-8. DDR SDRAM Control Configuration Register (DDR\_SDRAM\_CFG)**

Table 9-12 describes the DDR\_SDRAM\_CFG fields.

**Table 9-12. DDR\_SDRAM\_CFG Field Descriptions**

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.
2	ECC_EN	ECC enable. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and ECC_EN must be one to ensure an interrupt is generated. See <a href="#">Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."</a> 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs. Note that RD_EN and 2T_EN must not both be set at the same time.
4	—	Reserved
5–7	SDRAM_TYPE	Type of SDRAM device to be used. This field will be used when issuing the automatic hardware initialization sequence to DRAM via Mode Register Set and Extended Mode Register Set commands. Default value is 010 designating DDR1 SDRAM. 000–001 Reserved 010 DDR1 SDRAM 011 DDR2 SDRAM 100 Reserved 101 Reserved 110 Reserved 111 Reserved
8–9	—	Reserved
10	DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11	—	Reserved
12	32_BE	32-bit bus enable. 0 64-bit bus is used. 1 32-bit bus is used.
13	8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. <b>Note:</b> 8-beat bursts may be used by DDR1 (SDRAM_TYPE = 010), and only when using 32-bit bus mode (32_BE = 1); DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode

Table 9-12. DDR\_SDRAM\_CFG Field Descriptions (continued)

Bits	Name	Description
14	NCAP	Non-concurrent auto-precharge. Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, then this bit will need to be set if auto precharge is used. 0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
15	—	Reserved
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle. Note that RD_EN and 2T_EN must not both be set at the same time.
17–23	BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('x' denotes a don't care bit value. All unlisted field values are reserved.) 0000000 No external memory banks are interleaved 1000000 External memory banks 0 and 1 are interleaved 0100000 External memory banks 2 and 3 are interleaved 1100000 External memory banks 0 and 1 are interleaved together <b>and</b> banks 2 and 3 are interleaved together xx00100 External memory banks 0 through 3 are all interleaved together
24–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 will be used to capture DQ[0:31], DQS4 will be used to capture DQ[32:63] and DQS8 will be used to capture ECC[0:7].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] will be used to indicate the auto-precharge and precharge all commands. 1 MA[8] will be used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance will be used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in <a href="#">Section 19.4.1.21</a> , “DDR Control Driver Register (DDRCDR).” 0 I/O driver impedance will be configured to full strength. 1 I/O driver impedance will be configured to half strength.
29	—	Reserved

Table 9-12. DDR\_SDRAM\_CFG Field Descriptions (continued)

Bits	Name	Description
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller will not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software. 0 DDR controller will accept new transactions. 1 DDR controller will finish any remaining transactions, and then it will remain halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller will cycle through initialization routine based on SDRAM_TYPE 1 Initialization routine will be bypassed. Software is responsible for initializing memory through DDR_SDRAM_MODE2 register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller will not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled. See <a href="#">Section 9.4.1.15, "DDR Initialization Address (DDR_INIT_ADDR),"</a> for details on avoiding ECC errors in this mode.

### 9.4.1.8 DDR SDRAM Control Configuration 2 (DDR\_SDRAM\_CFG\_2)

The DDR SDRAM control configuration register 2, shown in [Figure 9-9](#), provides more control configuration for the DDR controller.

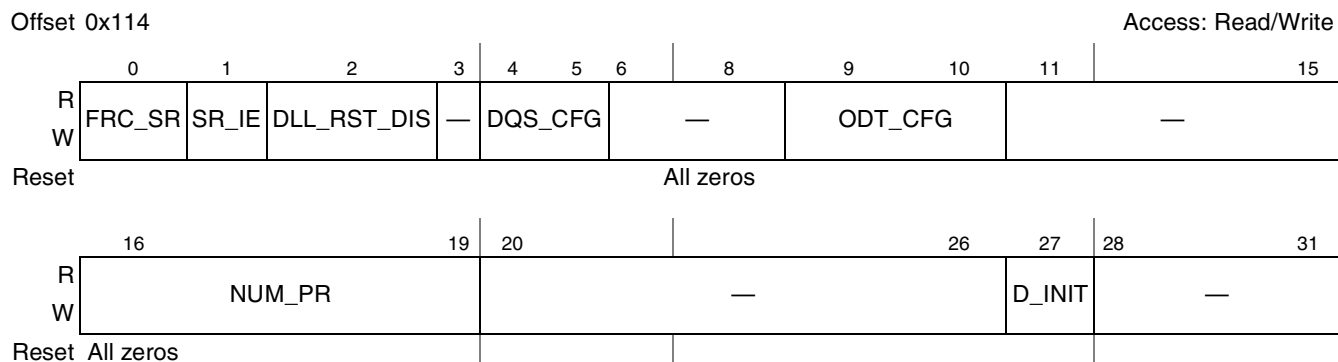


Figure 9-9. DDR SDRAM Control Configuration Register 2 (DDR\_SDRAM\_CFG\_2)

Table 9-13 describes the DDR\_SDRAM\_CFG\_2 fields.

**Table 9-13. DDR\_SDRAM\_CFG\_2 Field Descriptions**

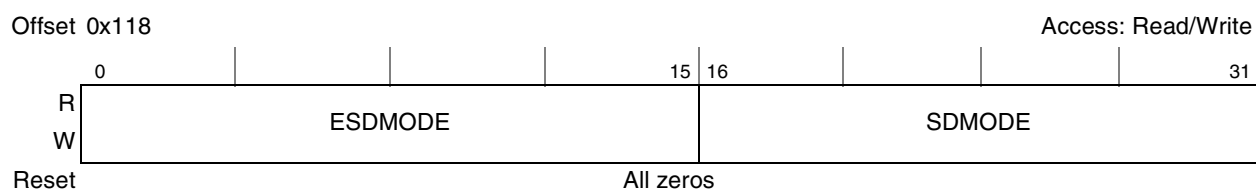
Bits	Name	Description
0	FRC_SR	Force self refresh 0 DDR controller will operate in normal mode. 1 DDR controller will enter self-refresh mode.
1	SR_IE	Self-refresh interrupt enable. The DDR controller can be placed into self refresh mode by forcing the PIC to assert IRQ_OUT. This is considered a 'panic interrupt' by the DDR controller, and it will enter self refresh as soon as possible. DDR_SDRAM_CFG[SREN] must also be set if the panic interrupt will be used. 0 DDR controller will not enter self-refresh mode if panic interrupt is asserted. 1 DDR controller will enter self-refresh mode if panic interrupt is asserted.
2	DLL_RST_DIS	DLL reset disable. The DDR controller will typically issue a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0 DDR controller will issue a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller will not issue a DLL reset to the DRAMs when exiting self refresh.
3	—	Reserved
4–5	DQS_CFG	DQS configuration 00 Only true DQS signals are used. 01 Differential DQS signals are used for DDR2 support. 10 Reserved 11 Reserved
6–8	—	Reserved
9–10	ODT_CFG	ODT configuration. This field defines how ODT will be driven to the on-chip IOs. See <a href="#">Section 19.4.1.21, “DDR Control Driver Register (DDRCDR)”</a> which defines the termination value that will be used. (DDR2-specific, must be cleared for DDR1) 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15	—	Reserved
16–19	NUM_PR	Number of posted refreshes. This will determine how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum $t_{ras}$ specification cannot be violated. For example, some DDR1 SDRAMs will not be able to use more than 3 posted refreshes because the required refresh interval could then exceed the maximum constraint for $t_{ras}$ . 0000 Reserved 0001 1 refresh will be issued at a time 0010 2 refreshes will be issued at a time 0011 3 refreshes will be issued at a time ... 1000 8 refreshes will be issued at a time 1001–1111 Reserved
20–26	—	Reserved, should be cleared.

**Table 9-13. DDR\_SDRAM\_CFG\_2 Field Descriptions (continued)**

Bits	Name	Description
27	D_INIT	<p>DRAM data initialization This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller will automatically initialize DRAM after it is enabled. This bit will be automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle.</p> <p>0 There is not data initialization in progress, and no data initialization is scheduled  1 The memory controller will initialize memory once it is enabled. This bit will remain asserted until the initialization is complete. The value in DDR_DATA_INIT register will be used to initialize memory.</p>
28–31	—	Reserved

### 9.4.1.9 DDR SDRAM Mode Configuration (DDR\_SDRAM\_MODE)

The DDR SDRAM mode configuration register, shown in [Figure 9-10](#), sets the values loaded into the DDR's mode registers.

**Figure 9-10. DDR SDRAM Mode Configuration Register (DDR\_SDRAM\_MODE)**

[Table 9-14](#) describes the DDR\_SDRAM\_MODE fields.

**Table 9-14. DDR\_SDRAM\_MODE Field Descriptions**

Bits	Name	Description
0–15	ESDMODE	<p>Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in <a href="#">Figure 9-10</a>, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0].</p>
16–31	SDMODE	<p>SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in <a href="#">Figure 9-10</a>, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].</p>

### 9.4.1.10 DDR SDRAM Mode 2 Configuration (DDR\_SDRAM\_MODE\_2)

The DDR SDRAM mode 2 configuration register, shown in Figure 9-11, sets the values loaded into the DDR’s extended mode 2 and 3 registers (for DDR2).

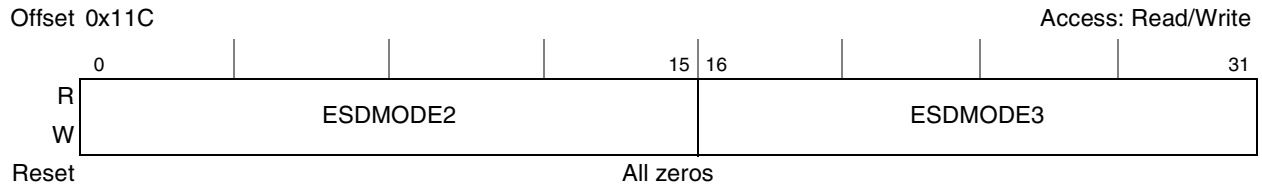


Figure 9-11. DDR SDRAM Mode 2 Configuration Register (DDR\_SDRAM\_MODE\_2)

Table 9-15 describes the DDR\_SDRAM\_MODE\_2 fields.

Table 9-15. DDR\_SDRAM\_MODE\_2 Field Descriptions

Bits	Name	Description
0–15	ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31	ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

### 9.4.1.11 DDR SDRAM Mode Control Register (DDR\_SDRAM\_MD\_CNTL)

The DDR SDRAM mode control register, shown in Figure 9-12, allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signals to a specific value

Table 9-16 describes the fields of this register. Table 9-17 shows the user how to set the fields of this register to accomplish the above tasks.

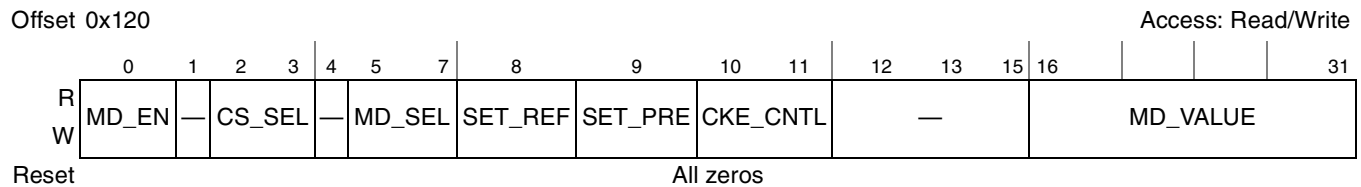


Figure 9-12. DDR SDRAM Mode Control Register (DDR\_SDRAM\_MD\_CNTL)



Table 9-16 describes the DDR\_SDRAM\_MD\_CNTL fields.

### NOTE

Note that MD\_EN, SET\_REF, and SET\_PRE are mutually exclusive; only one of these fields can be set at a time.

**Table 9-16. DDR\_SDRAM\_MD\_CNTL Field Descriptions**

Bits	Name	Description
0	MD_EN	<p>Mode enable. Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands:</p> <ul style="list-style-type: none"> <li>• MODE REGISTER SET</li> <li>• EXTENDED MODE REGISTER SET</li> <li>• EXTENDED MODE REGISTER SET 2</li> <li>• EXTENDED MODE REGISTER SET 3</li> </ul> <p>The specific command to be executed is selected by setting MD_SEL. In addition, the chip select must be chosen by setting CS_SEL. MD_EN is set by software and cleared by hardware once the command has been issued.</p> <p>0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.</p>
1	—	Reserved
2–3	CS_SEL	<p>Select chip select. Specifies the chip select that will be driven active due to any command forced by software in DDR_SDRAM_MD_CNTL.</p> <p>00 Chip select 0 is active 01 Chip select 1 is active 10 Chip select 2 is active 11 Chip select 3 is active</p>
4	—	Reserved
5–7	MD_SEL	<p>Mode register select. MD_SEL specifies one of the following:</p> <ul style="list-style-type: none"> <li>• During a mode select command, selects the SDRAM mode register to be changed</li> <li>• During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field.</li> <li>• During a refresh command, this field is ignored.</li> </ul> <p>Note that MD_SEL contains the value that will be presented onto the memory bank address pins (MBA<sub>n</sub>) of the DDR controller.</p> <p>000 MR 001 EMR 010 EMR2 011 EMR3</p>
8	SET_REF	<p>Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued.</p> <p>0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.</p>
9	SET_PRE	<p>Set precharge. Forces a precharge or precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued.</p> <p>0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.</p>

**Table 9-16. DDR\_SDRAM\_MD\_CNTL Field Descriptions (continued)**

Bits	Name	Description
10–11	CKE_CNTL	<p>Clock enable control. Allows software to globally clear or set all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value will continue to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller will continue to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (i.e., self refresh entry/exit, power down entry/exit).</p> <p>00 CKE signals are not forced by software.                      01 CKE signals are forced to a low value by software.                      10 CKE signals are forced to a high value by software.                      11 Reserved</p>
12–15	—	Reserved
16–31	MD_VALUE	<p>Mode register value. This field, which specifies the value that will be presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command.</p> <p>For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant:</p> <p>0 Issue a precharge command; MD_SEL selects the logical bank to be precharged                      1 Issue a precharge all command; all logical banks are precharged</p>

Table 9-17 shows how DDR\_SDRAM\_MD\_CNTL fields should be set for each of the tasks described above.

**Table 9-17. Settings of DDR\_SDRAM\_MD\_CNTL Fields**

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	—
SET_REF	0	1	0	—
SET_PRE	0	0	1	—
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See Table 9-16.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 9-16.	—
CKE_CNTL	0	0	0	See Table 9-16.

### 9.4.1.12 DDR SDRAM Interval Configuration (DDR\_SDRAM\_INTERVAL)

The DDR SDRAM interval configuration register, shown in [Figure 9-13](#), sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.



**Figure 9-13. DDR SDRAM Interval Configuration Register (DDR\_SDRAM\_INTERVAL)**

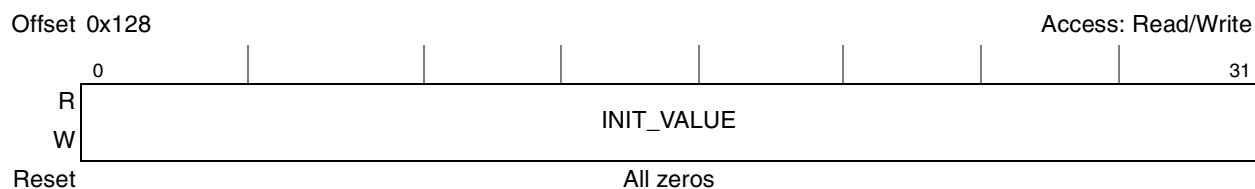
[Table 9-18](#) describes the DDR\_SDRAM\_INTERVAL fields.

**Table 9-18. DDR\_SDRAM\_INTERVAL Field Descriptions**

Bits	Name	Description
0–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes will not be issued when the REFINT is set to all 0s.
16–17	—	Reserved
18–31	BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

### 9.4.1.13 DDR SDRAM Data Initialization (DDR\_DATA\_INIT)

The DDR SDRAM data initialization register, shown in [Figure 9-14](#), provides the value that will be used to initialize memory if DDR\_SDRAM\_CFG2[D\_INIT] is set.



**Figure 9-14. DDR SDRAM Data Initialization Configuration Register (DDR\_DATA\_INIT)**

[Table 9-19](#) describes the DDR\_DATA\_INIT fields.

**Table 9-19. DDR\_DATA\_INIT Field Descriptions**

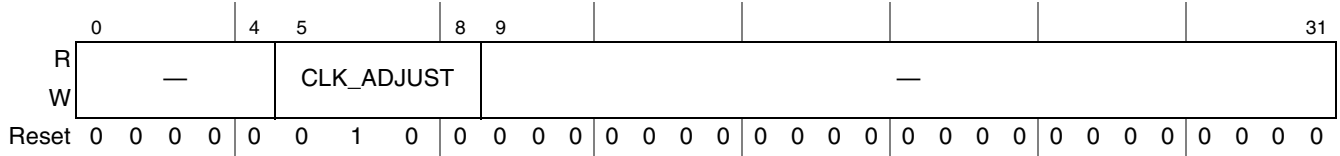
Bits	Name	Description
0–31	INIT_VALUE	Initialization value. Represents the value that DRAM will be initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

### 9.4.1.14 DDR SDRAM Clock Control (DDR\_SDRAM\_CLK\_CNTL)

The DDR SDRAM clock control configuration register, shown in Figure 9-15, provides a 1/8-cycle clock adjustment.

Offset 0x130

Access: Read/Write



**Figure 9-15. DDR SDRAM Clock Control Configuration Register (DDR\_SDRAM\_CLK\_CNTL)**

Table 9-20 describes the DDR\_SDRAM\_CLK\_CNTL fields.

**Table 9-20. DDR\_SDRAM\_CLK\_CNTL Field Descriptions**

Bits	Name	Description
0–4	—	Reserved
5–8	CLK_ADJUST	Clock adjust 0000 Clock will be launched aligned with address/command 0001 Clock will be launched 1/8 applied cycle after address/command 0010 Clock will be launched 1/4 applied cycle after address/command 0011 Clock will be launched 3/8 applied cycle after address/command 0100 Clock will be launched 1/2 applied cycle after address/command 0101 Clock will be launched 5/8 applied cycle after address/command 0110 Clock will be launched 3/4 applied cycle after address/command 0111 Clock will be launched 7/8 applied cycle after address/command 1000 Clock will be launched 1 applied cycle after address/command 1001–1111 Reserved
9–31	—	Reserved

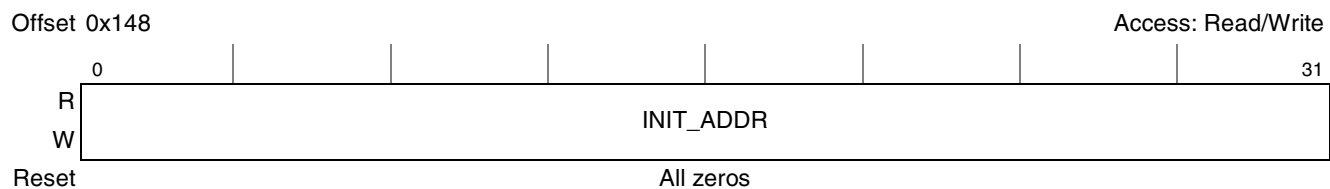
### 9.4.1.15 DDR Initialization Address (DDR\_INIT\_ADDR)

The DDR SDRAM initialization address register, shown in Figure 9-16, provides the address that will be used for the data strobe to data skew adjustment and automatic  $\overline{\text{CAS}}$  to preamble calibration after POR.

#### NOTE

After the skew adjustment, this address will contain bad ECC data. This is not important at POR, as all of memory should be subsequently initialized if ECC is enabled (either by software or through the use of DDR\_SDRAM\_CFG\_2[D\_INIT]).

If an  $\overline{\text{HRESET}}$  has been issued after the DRAM is in self-refresh mode, however, memory is not initialized, so this address should be written to using an 8- or 32-byte transaction to avoid possible ECC errors if this address could later be accessed.



**Figure 9-16. DDR Initialization Address Configuration Register (DDR\_INIT\_ADDR)**

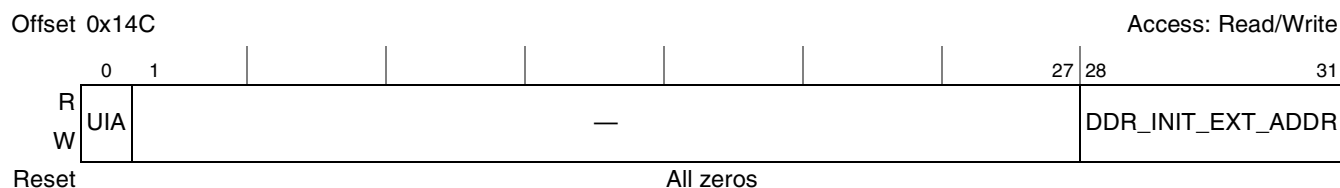
Table 9-21 describes the DDR\_INIT\_ADDR fields.

**Table 9-21. DDR\_INIT\_ADDR Field Descriptions**

Bits	Name	Description
0–31	INIT_ADDR	Initialization address. Represents the address that will be used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at POR. This address will be written to during the initialization sequence.

#### 9.4.1.16 DDR Initialization Enable Extended Address (DDR\_INIT\_EXT\_ADDR)

The DDR SDRAM initialization extended address register, shown in Figure 9-17, provides the extended address that will be used for the data strobe to data skew adjustment and automatic  $\overline{\text{CAS}}$  to preamble calibration after POR.



**Figure 9-17. DDR Initialization Extended Address Configuration Register (DDR\_INIT\_EXT\_ADDR)**

Table 9-22 describes the DDR\_INIT\_EXT\_ADDR fields.

**Table 9-22. DDR\_INIT\_EXT\_ADDR Field Descriptions**

Bits	Name	Description
0	UIA	Use initialization address. 0 Use the default address for training sequence as calculated by the controller. This will be the first valid address in the first enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR and DDR_INIT_EXT_ADDR.
1–27	—	Reserved, should be cleared.
28–31	INIT_EXT_ADDR	Initialization extended address. Represents the extended address that will be used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration at POR. This extended address will be written to during the initialization sequence.

### 9.4.1.17 DDR IP Block Revision 1 (DDR\_IP\_REV1)

The DDR IP block revision 1 register, shown in Figure 9-18, provides read-only fields with the IP block ID, along with major and minor revision information.

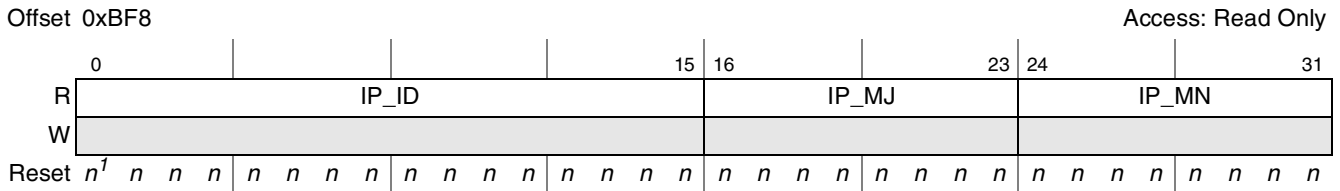


Figure 9-18. DDR IP Block Revision 1 (DDR\_IP\_REV1)

<sup>1</sup> For reset values, see Table 9-23.

Table 9-23 describes the DDR\_IP\_REV1 fields.

Table 9-23. DDR\_IP\_REV1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID. For the DDR controller, this value is 0x0002.
16–23	IP_MJ	Major revision. This is currently set to 0x02.
24–31	IP_MN	Minor revision. This is currently set to 0x00.

### 9.4.1.18 DDR IP Block Revision 2 (DDR\_IP\_REV2)

The DDR IP block revision 2 register, shown in Figure 9-19, provides read-only fields with the IP block integration and configuration options.

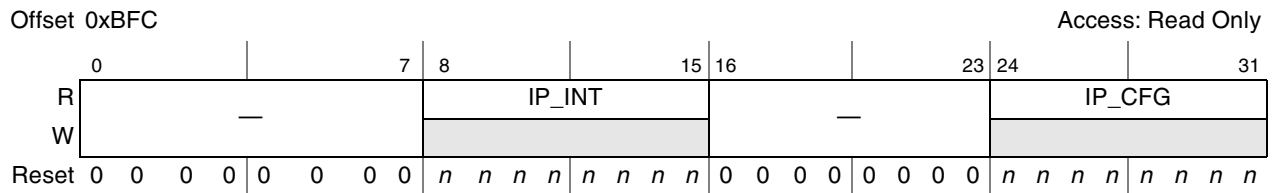


Figure 9-19. DDR IP Block Revision 2 (DDR\_IP\_REV2)

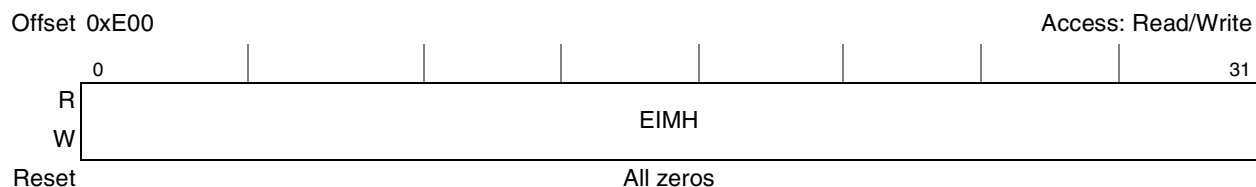
Table 9-24 describes the DDR\_IP\_REV2 fields.

Table 9-24. DDR\_IP\_REV2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

### 9.4.1.19 Memory Data Path Error Injection Mask High (DATA\_ERR\_INJECT\_HI)

The memory data path error injection mask high register is shown in [Figure 9-20](#).



**Figure 9-20. Memory Data Path Error Injection Mask High Register (DATA\_ERR\_INJECT\_HI)**

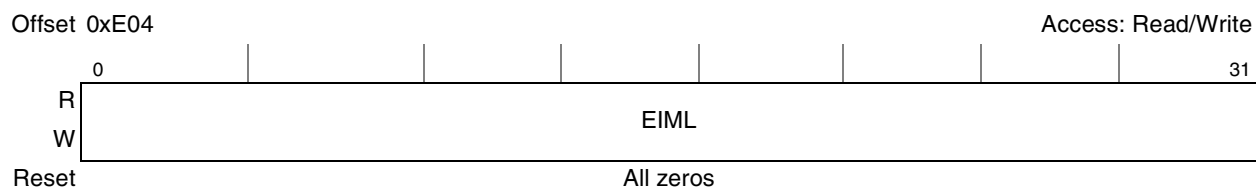
[Table 9-25](#) describes the DATA\_ERR\_INJECT\_HI fields.

**Table 9-25. DATA\_ERR\_INJECT\_HI Field Descriptions**

Bits	Name	Description
0–31	EIMH	Error injection mask high data path. Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

### 9.4.1.20 Memory Data Path Error Injection Mask Low (DATA\_ERR\_INJECT\_LO)

The memory data path error injection mask low register is shown in [Figure 9-21](#).



**Figure 9-21. Memory Data Path Error Injection Mask Low Register (DATA\_ERR\_INJECT\_LO)**

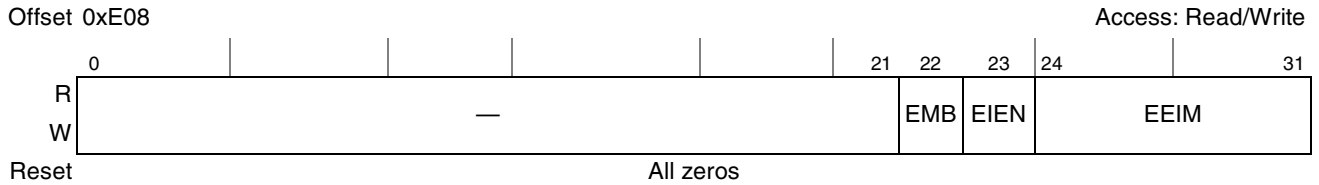
[Table 9-26](#) describes the DATA\_ERR\_INJECT\_LO fields.

**Table 9-26. DATA\_ERR\_INJECT\_LO Field Descriptions**

Bits	Name	Description
0–31	EIML	Error injection mask low data path. Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

### 9.4.1.21 Memory Data Path Error Injection Mask ECC (ERR\_INJECT)

The memory data path error injection mask ECC register, shown in [Figure 9-22](#), sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.



**Figure 9-22. Memory Data Path Error Injection Mask ECC Register (ERR\_INJECT)**

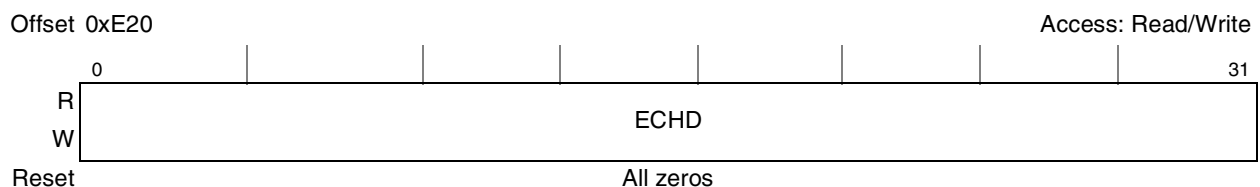
[Table 9-27](#) describes the ERR\_INJECT fields.

**Table 9-27. ERR\_INJECT Field Descriptions**

Bits	Name	Description
0–21	—	Reserved
22	EMB	ECC mirror byte 0 Mirror byte functionality disabled. 1 Mirror the most significant data path byte onto the ECC byte.
23	EIEN	Error injection enable 0 Error injection disabled. 1 Error injection enabled. This applies to the data mask bits, the ECC mask bits, and the ECC mirror bit. Note that error injection should not be enabled until the memory controller has been enabled via DDR_SDRAM_CFG[MEM_EN].
24–31	EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

### 9.4.1.22 Memory Data Path Read Capture High (CAPTURE\_DATA\_HI)

The memory data path read capture high register, shown in [Figure 9-23](#), stores the high word of the read data path during error capture.



**Figure 9-23. Memory Data Path Read Capture High Register (CAPTURE\_DATA\_HI)**

[Table 9-28](#) describes the CAPTURE\_DATA\_HI fields.

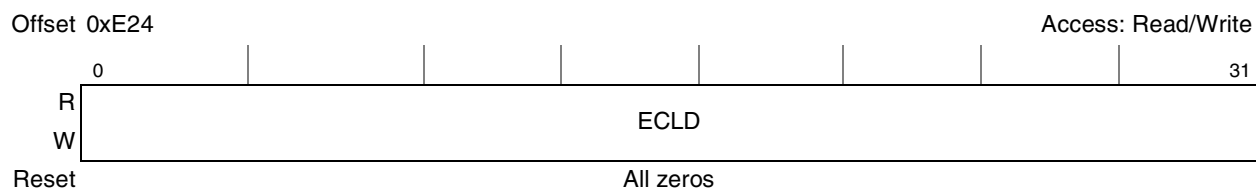
**Table 9-28. CAPTURE\_DATA\_HI Field Descriptions**

Bits	Name	Description
0–31	ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.



### 9.4.1.23 Memory Data Path Read Capture Low (CAPTURE\_DATA\_LO)

The memory data path read capture low register, shown in [Figure 9-24](#), stores the low word of the read data path during error capture.



**Figure 9-24. Memory Data Path Read Capture Low Register (CAPTURE\_DATA\_LO)**

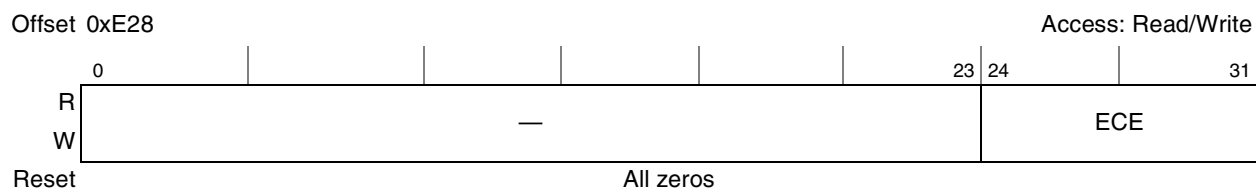
[Table 9-29](#) describes the CAPTURE\_DATA\_LO fields.

**Table 9-29. CAPTURE\_DATA\_LO Field Descriptions**

Bits	Name	Description
0–31	ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

### 9.4.1.24 Memory Data Path Read Capture ECC (CAPTURE\_ECC)

The memory data path read capture ECC register, shown in [Figure 9-25](#), stores the ECC syndrome bits that were on the data bus when an error was detected.



**Figure 9-25. Memory Data Path Read Capture ECC Register (CAPTURE\_ECC)**

[Table 9-30](#) describes the CAPTURE\_ECC fields.

**Table 9-30. CAPTURE\_ECC Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected.

### 9.4.1.25 Memory Error Detect (ERR\_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR\_DISABLE, the corresponding error is never detected or captured in ERR\_DETECT.

ERR\_DETECT is shown in [Figure 9-26](#).

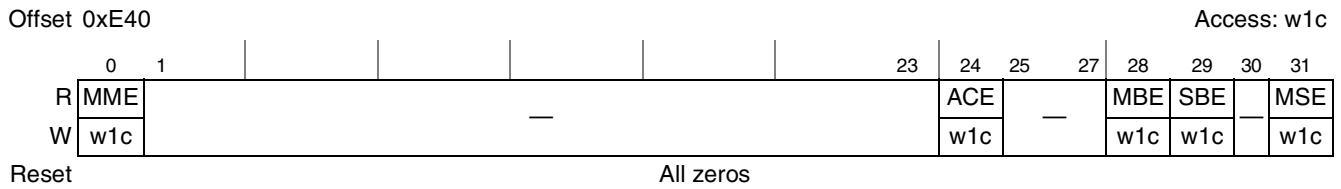


Figure 9-26. Memory Error Detect Register (ERR\_DETECT)

Table 9-31 describes the ERR\_DETECT fields.

Table 9-31. ERR\_DETECT Field Descriptions

Bits	Name	Description
0	MME	Multiple memory errors. This bit is cleared by software writing a 1. 0 Multiple memory errors of the same type were not detected. 1 Multiple memory errors of the same type were detected.
1–23	—	Reserved
24	ACE	Automatic calibration error. This bit is cleared by software writing a 1. 0 An automatic calibration error has not been detected. 1 An automatic calibration error has been detected.
25–27	—	Reserved
28	MBE	Multiple-bit error. This bit is cleared by software writing a 1. 0 A multiple-bit error has not been detected. 1 A multiple-bit error has been detected.
29	SBE	Single-bit ECC error. This bit is cleared by software writing a 1. 0 The number of single-bit ECC errors detected has not crossed the threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30	—	Reserved
31	MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

**9.4.1.26 Memory Error Disable (ERR\_DISABLE)**

The memory error disable register, shown in [Figure 9-27](#), allows selective disabling of the DDR controller’s error detection circuitry. Disabled errors are not detected or reported.

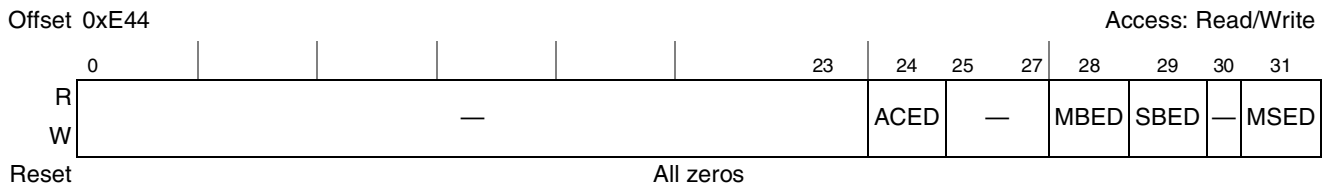


Figure 9-27. Memory Error Disable Register (ERR\_DISABLE)

Table 9-32 describes the ERR\_DISABLE fields.

Table 9-32. ERR\_DISABLE Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	ACED	Automatic calibration error disable 0 Automatic calibration errors are enabled. 1 Automatic calibration errors are disabled.
25–27	—	Reserved
28	MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. Note that uncorrectable read errors cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED and ERR_INT_EN[MBEE] must be zero and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

#### 9.4.1.27 Memory Error Interrupt Enable (ERR\_INT\_EN)

The memory error interrupt enable register, shown in Figure 9-28, enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

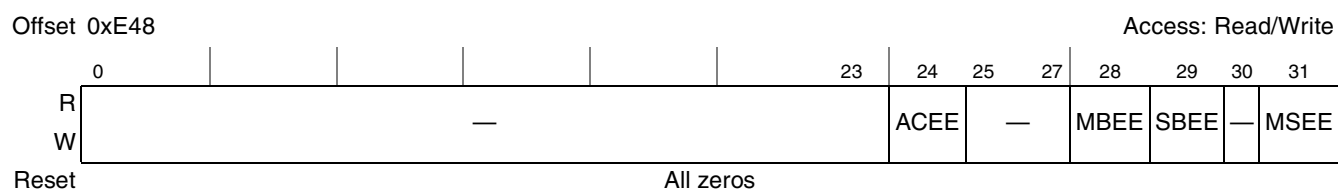


Figure 9-28. Memory Error Interrupt Enable Register (ERR\_INT\_EN)

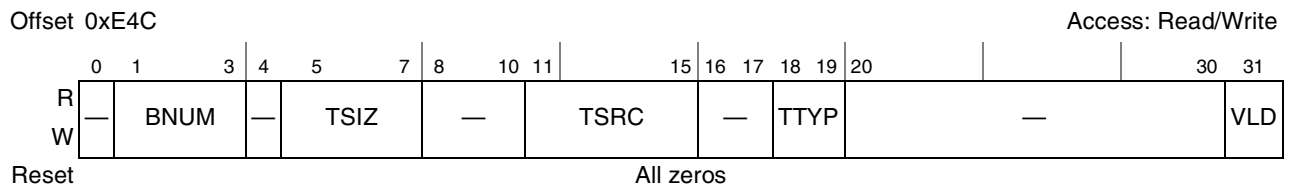
Table 9-33 describes the ERR\_INT\_EN fields.

**Table 9-33. ERR\_INT\_EN Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24	ACEE	Automatic calibration error interrupt enable 0 Automatic calibration errors cannot generate interrupts. 1 Automatic calibration errors generate interrupts.
25–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBEE and ERR_DISABLE[MBED] must be zero and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”</a> 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29	SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

### 9.4.1.28 Memory Error Attributes Capture (CAPTURE\_ATTRIBUTES)

The memory error attributes capture register, shown in [Figure 9-29](#), sets attributes for errors including type, size, source, and others.



**Figure 9-29. Memory Error Attributes Capture Register (CAPTURE\_ATTRIBUTES)**

Table 9-34 describes the CAPTURE\_ATTRIBUTES fields.

**Table 9-34. CAPTURE\_ATTRIBUTES Field Descriptions**

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4	—	Reserved

Table 9-34. CAPTURE\_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
5–7	TSIZ	Transaction size for the error. Captures the transaction size in double words. 000 4 double words 001 1 double word 010 2 double words 011 3 double words Others Reserved
8–10	—	Reserved
11–15	TSRC	Transaction source for the error 00000 PCI interface 00001 PCI Express interface 2 00010 PCI Express interface 1 00011 PCI Express interface 3 00100 Local Bus 00101–00111 Reserved 01000 Configuration space 01001 Reserved 01010 Boot sequencer 01011–01110 Reserved 01111 DDR controller 10000 Processor (instruction) 10001 Processor (data) 10010 Reserved 10011 Reserved 10100 Reserved 10101 DMA 10110 Reserved 10111 SAP 11000 eTSEC 1 11001 Reserved 11010 eTSEC 3 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved
16–17	—	Reserved
18–19	TTYP	Transaction type for the error 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VLD	Valid. Set as soon as valid information is captured in the error capture registers.

### 9.4.1.29 Memory Error Address Capture (CAPTURE\_ADDRESS)

The memory error address capture register, shown in Figure 9-30, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

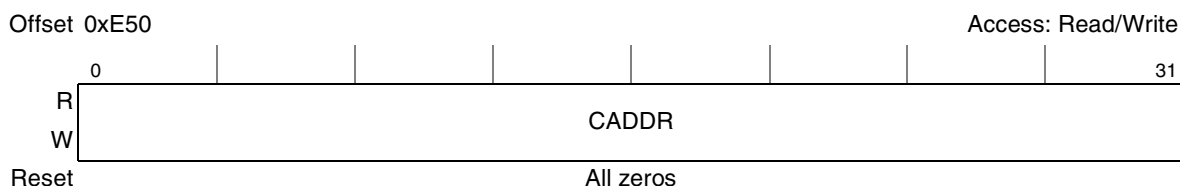


Figure 9-30. Memory Error Address Capture Register (CAPTURE\_ADDRESS)

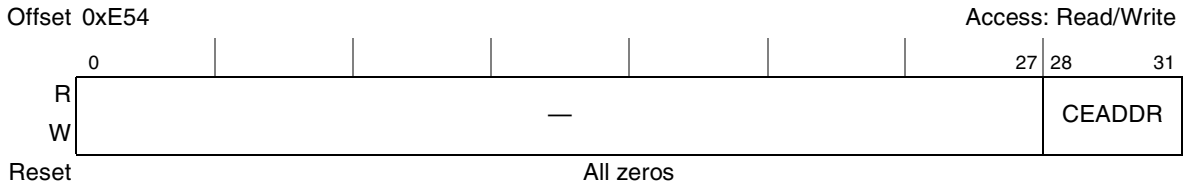
Table 9-35 describes the CAPTURE\_ADDRESS fields.

**Table 9-35. CAPTURE\_ADDRESS Field Descriptions**

Bits	Name	Description
0–31	CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

### 9.4.1.30 Memory Error Extended Address Capture (CAPTURE\_EXT\_ADDRESS)

The memory error extended address capture register, shown in Figure 9-31, holds the four most significant transaction bits when an error is detected.



**Figure 9-31. Memory Error Extended Address Capture Register (CAPTURE\_EXT\_ADDRESS)**

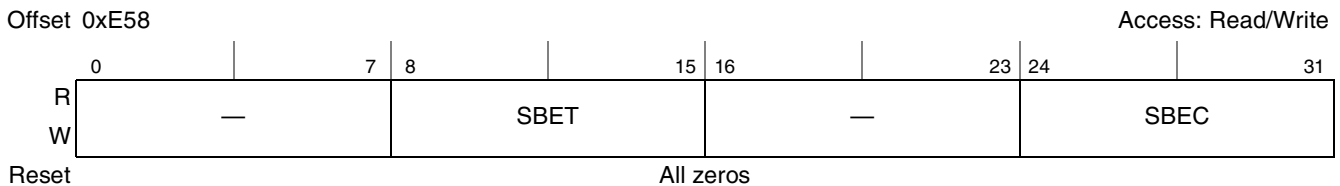
Table 9-36 describes the CAPTURE\_EXT\_ADDRESS fields.

**Table 9-36. CAPTURE\_EXT\_ADDRESS Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	CEADDR	Captured extended address. Captures the 4 msbs of the transaction address when an error is detected

### 9.4.1.31 Single-Bit ECC Memory Error Management (ERR\_SBE)

The single-bit ECC memory error management register, shown in Figure 9-32, stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.



**Figure 9-32. Single-Bit ECC Memory Error Management Register (ERR\_SBE)**

Table 9-37 describes the ERR\_SBE fields.

**Table 9-37. ERR\_SBE Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–15	SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.
16–23	—	Reserved
24–31	SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and a machine check or critical interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

## 9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

Figure 9-33 is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40bit wide memory. Bank sizes up to 4 Gbytes are supported, providing up to a maximum of 16 Gbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32-bit data bus, detects all double-bit errors within the 64- or 32-bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR\_SDRAM\_INTERVAL[BSTOPRE].

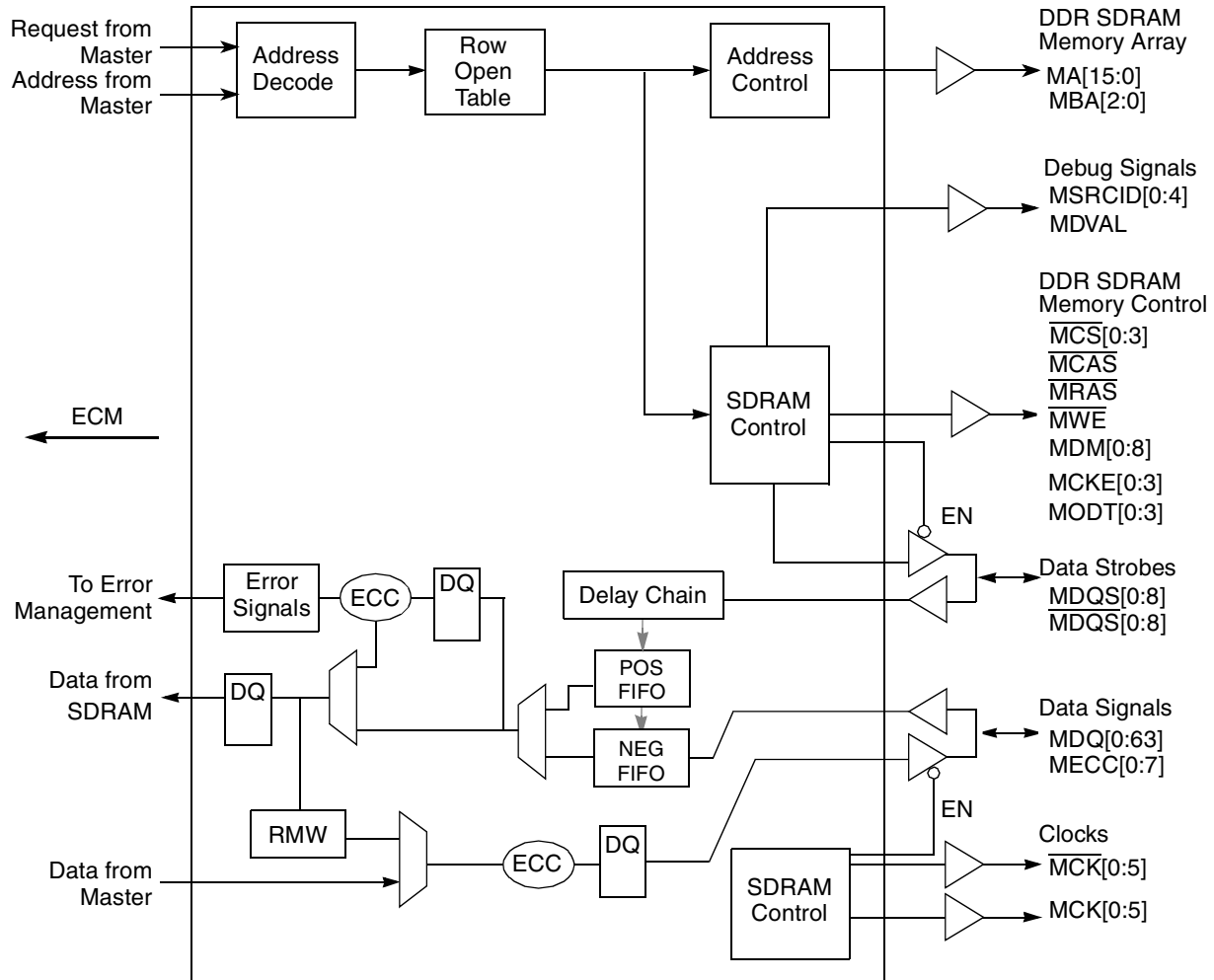


Figure 9-33. DDR Memory Controller Block Diagram

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (4 or 8) in a programmed sequence. Accesses to closed pages start with the registration of an ACTIVE command followed by a READ or WRITE. (Accessing open pages does not require an ACTIVE command.) The address bits registered coincident with the activate command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.



The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

Figure 9-34 shows an example DDR SDRAM configuration with four logical banks.

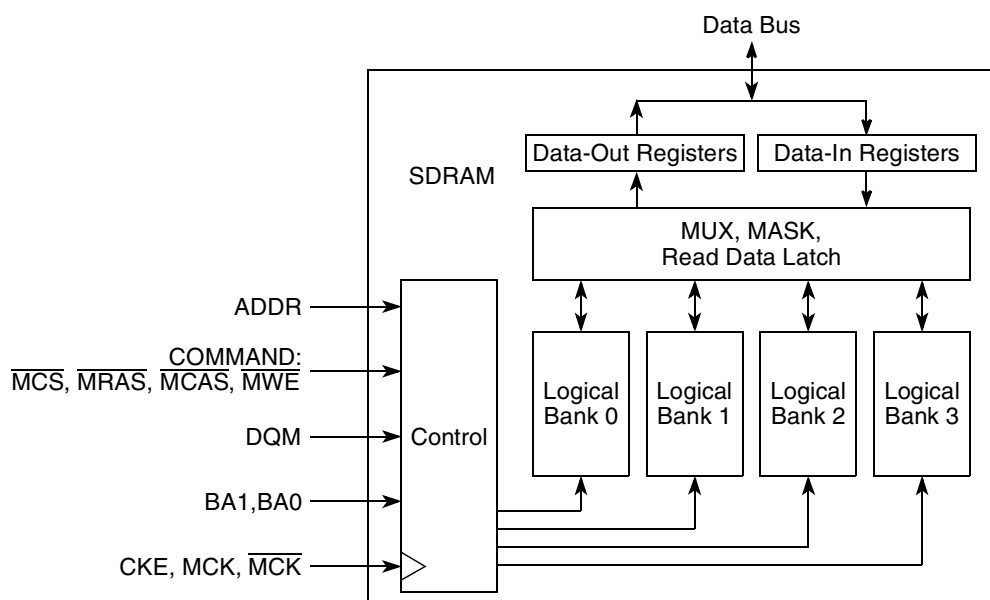


Figure 9-34. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-35 shows some typical signal connections.

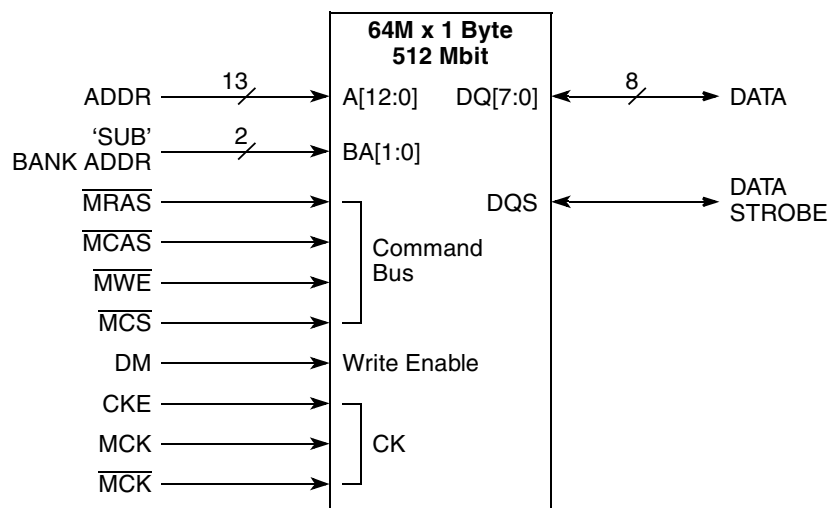
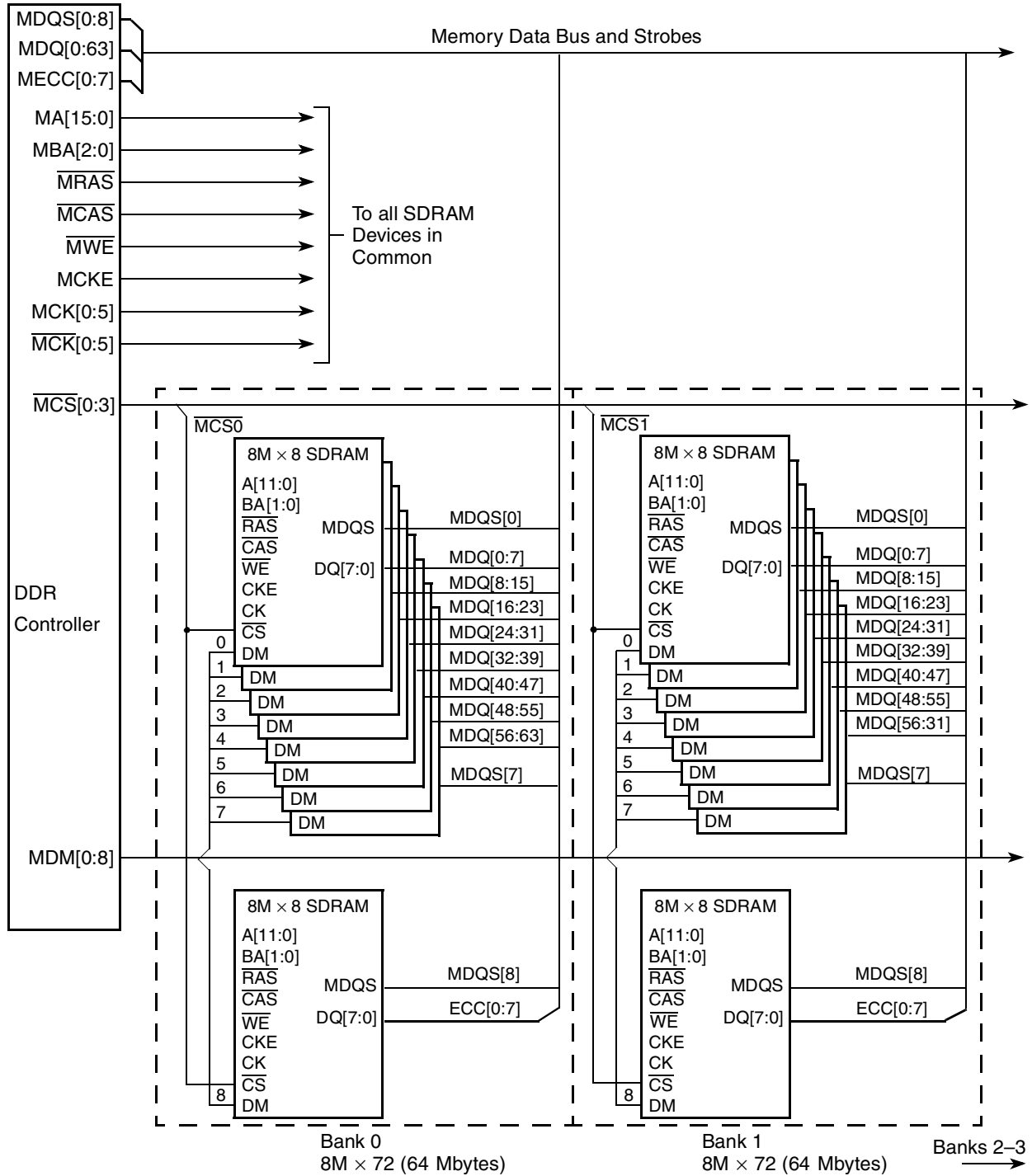


Figure 9-35. Typical DDR SDRAM Interface Signals

Figure 9-36 shows an example DDR SDRAM configuration with four physical banks each comprised of nine  $8\text{M} \times 8$  DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 12 bits.



1. All signals are connected in common (in parallel) except for  $\overline{MCS}[0:3]$ ,  $\overline{MCK}[0:5]$ ,  $\overline{MDM}[0:8]$ , and the data bus signals.
2. Each of the  $\overline{MCS}[0:3]$  signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4.  $\overline{MCK}[0:5]$  may be apportioned among all memory devices. Complementary bus is not shown.

**Figure 9-36. Example 256-Mbyte DDR SDRAM Configuration With ECC**

Section 9.5.12, “Error Management,” explains how the DDR memory controller handles errors.

## 9.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Sixteen multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 4 Gbits. Four chip select ( $\overline{CS}$ ) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 Mbytes to 4 Gbytes. The physical banks can be constructed using x8, x16, or x32 memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

### NOTE

An 8-bit DDR SDRAM device has a DQM signal and eight data signals (DQ[0:7]). A 16-bit DDR SDRAM device has two DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection.

Table 9-38 shows the DDR memory controller's relationships between data byte lane0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

**Table 9-38. Byte Lane to Data Relationship**

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 64-Bit Mode
0 (MSB)	MDM[0]	MDQS[0]	MDQ[0:7]
1	MDM[1]	MDQS[1]	MDQ[8:15]
2	MDM[2]	MDQS[2]	MDQ[16:23]
3	MDM[3]	MDQS[3]	MDQ[24:31]
4	MDM[4]	MDQS[4]	MDQ[32:39]
5	MDM[5]	MDQS[5]	MDQ[40:47]
6	MDM[6]	MDQS[6]	MDQ[48:55]
7 (LSB)	MDM[7]	MDQS[7]	MDQ[56:63]

### 9.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 1616 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 16 row address bits, plus 2 or 3 logical bank-select bits and from 8–11 column address bits.

Table 9-39 describe DDR SDRAM device configurations supported by the DDR memory controller.

**NOTE**

DDR SDRAM is limited to 30 total address bits.

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits x 8	12 x 9 x 2	64 Mbytes	256 Mbytes
64 Mbits <sup>1</sup>	4 Mbits x 16	12 x 8 x 2	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	13 x 11 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 11 x 2	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	14 x 10 x 2	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 11 x 2	2 Gbytes	8 Gbytes
2 Gbits	128 Mbits x 16	15 x 10 x 2	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 11 x 2	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	16 x 10 x 2	2 Gbytes	8 Gbytes

<sup>1</sup> This configuration is not supported in 16-bit bus mode.

**Table 9-39. Supported DDR2 SDRAM Device Configurations**

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes
2 Gbits	128 Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in [Section 9.5.12, “Error Management.”](#)

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the DDR memory controller to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate  $\overline{MCS}_n$  signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

### 9.5.2 DDR SDRAM Address Multiplexing

Table 9-40, Table 9-41 Table 9-42, and Table 9-43 show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[15:0] use MA[15] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA[10].

**Table 9-40. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled**

Row x Col	msb	Address from Core Master																															lsb		
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35				
16 x 11 x 2	$\overline{MRAS}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																		1	0															
	$\overline{MCAS}$																				11	9	8	7	6	5	4	3	2	1	0				
16 x 10 x 2	$\overline{MRAS}$		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0														
	$\overline{MCAS}$																					9	8	7	6	5	4	3	2	1	0				
15 x 11 x 2	$\overline{MRAS}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0														
	$\overline{MCAS}$																					11	9	8	7	6	5	4	3	2	1	0			
15 x 10 x 2	$\overline{MRAS}$				14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																				1	0													
	$\overline{MCAS}$																						9	8	7	6	5	4	3	2	1	0			
14 x 11 x 2	$\overline{MRAS}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																				1	0													
	$\overline{MCAS}$																						11	9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	$\overline{MRAS}$						13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																					1	0												
	$\overline{MCAS}$																							9	8	7	6	5	4	3	2	1	0		
13 x 11 x 2	$\overline{MRAS}$							12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																					1	0												
	$\overline{MCAS}$																							11	9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{MRAS}$								12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																						1	0											
	$\overline{MCAS}$																								9	8	7	6	5	4	3	2	1	0	

Table 9-40. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																															lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35		
13 x 9 x 2	MRAS						12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0												
	MCAS																						8	7	6	5	4	3	2	1	0		
12 x 10 x 2	MRAS						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0		
12 x 9 x 2	MRAS						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																						8	7	6	5	4	3	2	1	0		
12 x 8 x 2	MRAS						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																				1	0											
	MCAS																							7	6	5	4	3	2	1	0		

Table 9-41. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																															lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34-35	
16 x 11 x 2	MRAS		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																					11	9	8	7	6	5	4	3	2	1	0	
16 x 10 x 2	MRAS		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0		
15 x 11 x 2	MRAS		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			1	0												
	MCAS																					11	9	8	7	6	5	4	3	2	1	0	
15 x 10 x 2	MRAS		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0		
14 x 11 x 2	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0												
	MCAS																					11	9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0		

Table 9-41. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																																lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34-35		
13 x 11 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0														
	MCAS																				11	9	8	7	6	5	4	3	2	1	0			
13 x 10 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0														
	MCAS																					9	8	7	6	5	4	3	2	1	0			
13 x 9 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			1	0													
	MCAS																						8	7	6	5	4	3	2	1	0			
12 x 10 x 2	MRAS					11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
12 x 9 x 2	MRAS					11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																				1	0												
	MCAS																						8	7	6	5	4	3	2	1	0			
12 x 8 x 2	MRAS					11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																					1	0											
	MCAS																							7	6	5	4	3	2	1	0			

Table 9-42. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																																lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35			
16 x 10 x 3	MRAS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																		2	1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
15 x 10 x 3	MRAS		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																		2	1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
14 x 10 x 3	MRAS			13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																		2	1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
14 x 10 x 2	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0														
	MCAS																					9	8	7	6	5	4	3	2	1	0			

Table 9-42. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																															Isb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35		
13 x 10 x 3	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																	2	1	0													
	MCAS																				9	8	7	6	5	4	3	2	1	0			
13 x 10 x 2	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0													
	MCAS																				9	8	7	6	5	4	3	2	1	0			
13 x 9 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																					8	7	6	5	4	3	2	1	0			

Table 9-43. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																															Isb	
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34-35		
16 x 10 x 3	MRAS		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		2	1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
15 x 10 x 3	MRAS			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			2	1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0			
14 x 10 x 3	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			2	1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0			
14 x 10 x 2	MRAS					13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																				1	0												
	MCAS																					9	8	7	6	5	4	3	2	1	0			
13 x 10 x 3	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																				2	1	0											
	MCAS																						9	8	7	6	5	4	3	2	1	0		



**Table 9-43. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)**

Row x Col	msb	Address from Core Master																																lsb
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34-35			
13 x 10 x 2	MRAS						12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0													
	MCAS																						9	8	7	6	5	4	3	2	1	0		
13 x 9 x 2	MRAS						12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																					1	0											
	MCAS																							8	7	6	5	4	3	2	1	0		

Chip select interleaving is supported for the memory controller, and is programmed in DDR\_SDRAM\_CFG[BA\_INTLV\_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 9-44 illustrates examples of address decode when interleaving between two chip selects, and Table 9-45 shows examples of address decode when interleaving between four chip selects.

**Table 9-44. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Two Banks**

Row x Col	msb	Address from Core Master																																lsb
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35				
14 x 10 x 3	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																			CS SEL	2	1	0											
	MCAS																						9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	MRAS			13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																					CS SEL	1	0										
	MCAS																							9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS			12	11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																						CS SEL	2	1	0								
	MCAS																							9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																							CS SEL	1	0								
	MCAS																								9	8	7	6	5	4	3	2	1	0

**Table 9-45. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks**

Row x Col	msb	Address from Core Master																																lsb
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33-35				
14 x 10 x 3	MRAS	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																		
	MBA																2	1	0															
	MCAS																			9	8	7	6	5	4	3	2	1	0					
14 x 10 x 2	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																	
	MBA																	1	0															
	MCAS																			9	8	7	6	5	4	3	2	1	0					
13 x 10 x 3	MRAS		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																		
	MBA																	2	1	0														
	MCAS																			9	8	7	6	5	4	3	2	1	0					
13 x 10 x 2	MRAS		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																		
	MBA																	1	0															
	MCAS																			9	8	7	6	5	4	3	2	1	0					

### 9.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR2 or DDR modes.

All read or write accesses to DDR SDRAM are performed by the DDR memory controller using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled using both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

The following DDR SDRAM interface commands (summarized in [Table 9-46](#)) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.

- **Write**—Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.
- **Refresh** (similar to  $\overline{\text{MCAS}}$  before  $\overline{\text{MRAS}}$ )—Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- **Mode register set** (for configuration)—Allows setting of DDR SDRAM options. These options are:  $\overline{\text{MCAS}}$  latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length.  $\overline{\text{MCAS}}$  latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide  $\overline{\text{MCAS}}$  latency {1,2,3}, some provide  $\overline{\text{MCAS}}$  latency {1,2,3,4,5}, etc.). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR1 memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data,  $\overline{\text{MCAS}}$  latency, burst length, and burst type, are set by software in `DDR_SDRAM_MODE[SDMODE]` and transferred to the SDRAM array by the DDR memory controller after `DDR_SDRAM_CFG[MEM_EN]` is set. If `DDR_SDRAM_CFG[BI]` is set to bypass the automatic initialization, then the MODE registers can be configured through software via use of the `DDR_SDRAM_MD_CNTL` register.
- **Self refresh** (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller will place all logical banks in a precharged state.

Table 9-46. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto-precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column

Table 9-46. DDR SDRAM Command Table (continued)

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Write with auto-precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

## 9.5.4 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

### NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in Table 9-47 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-47. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as $t_{\text{RRD}}$ .
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as $t_{\text{RAS}}$ .
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as $t_{\text{RCD}}$ .
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge $n$ , and the read latency is $m$ clocks, the data is available nominally coincident with clock edge $n + m$ .
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as $t_{\text{RP}}$ .

Table 9-47. DDR SDRAM Interface Timing Intervals (continued)

Timing Intervals	Definition
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[ <i>NUM_PR</i> ], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as $t_{RP}$ .
REFREC	The number of clock cycles from the refresh command until an activate command is allowed. This can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in nanoseconds.
WR_DATA_DELAY	Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.
WRREC	The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as $t_{WR}$ .
WRTORD	Last write pair to read command. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank as $t_{WTR}$ .

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING\_CFG\_0, TIMING\_CFG\_1, TIMING\_CFG\_2, and TIMING\_CFG\_3 registers as described in Section 9.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING\_CFG\_0),” Section 9.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING\_CFG\_1),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING\_CFG\_2),” and Section 9.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING\_CFG\_3),”) and be kept in the DDR memory controller configuration register space.

The following figures show SDRAM timing for various types of accesses. System software is responsible (at reset) for optimally configuring SDRAM timing parameters. The programmable timing parameters apply to both read and write timing configuration. The configuration process must be completed and the DDR SDRAM initialized before any accesses to SDRAM are attempted.

Figure 9-37 through Figure 9-40 show DDR SDRAM timing for various types of accesses; see Figure 9-37 for a single-beat read operation, Figure 9-38 for a single-beat write operation, and Figure 9-40 for a burst-write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK\_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle (for DDR1).

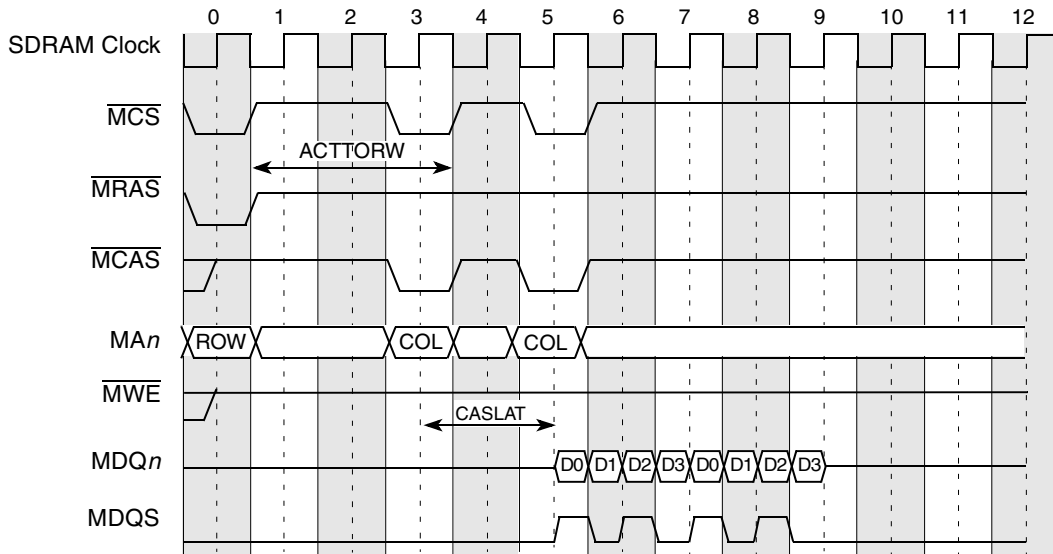


Figure 9-37. DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2

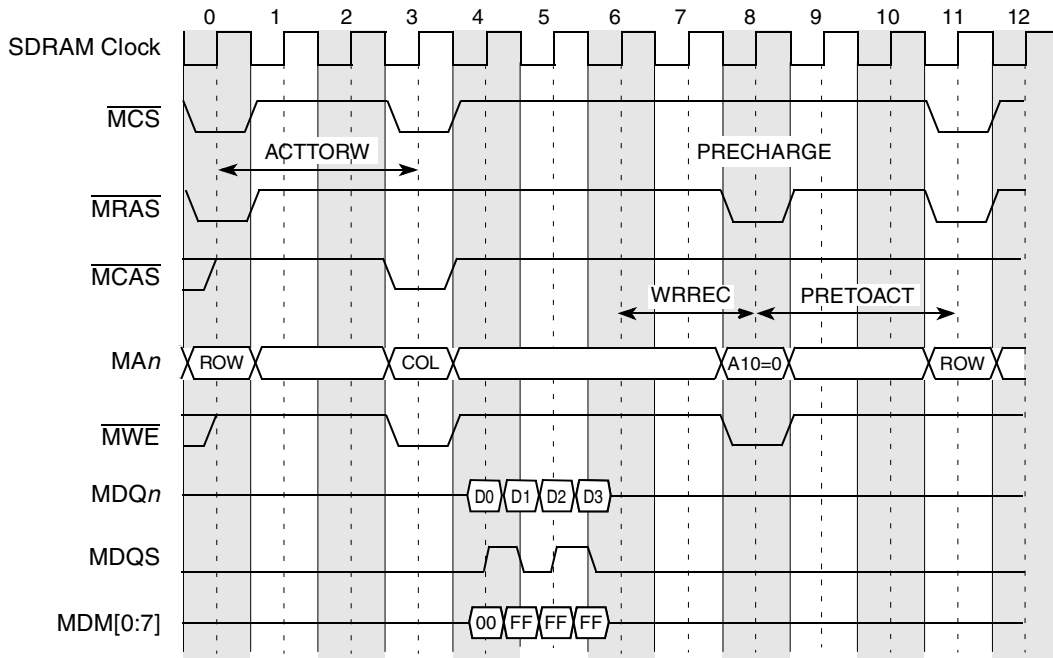


Figure 9-39. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

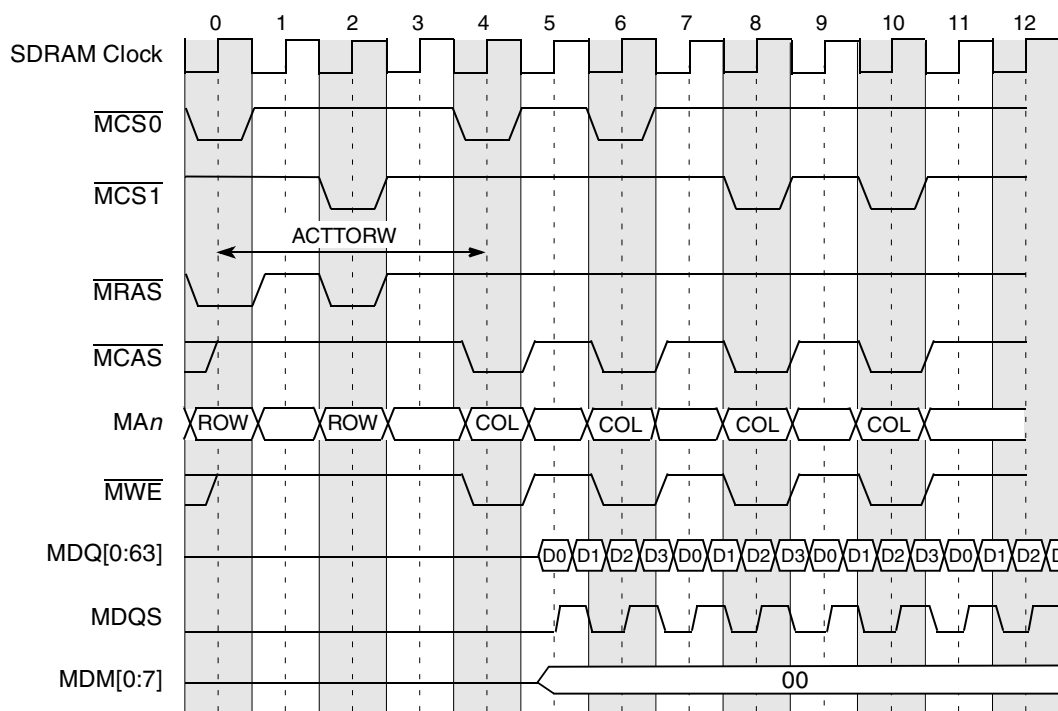


Figure 9-40. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

#### 9.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/MCK̄ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

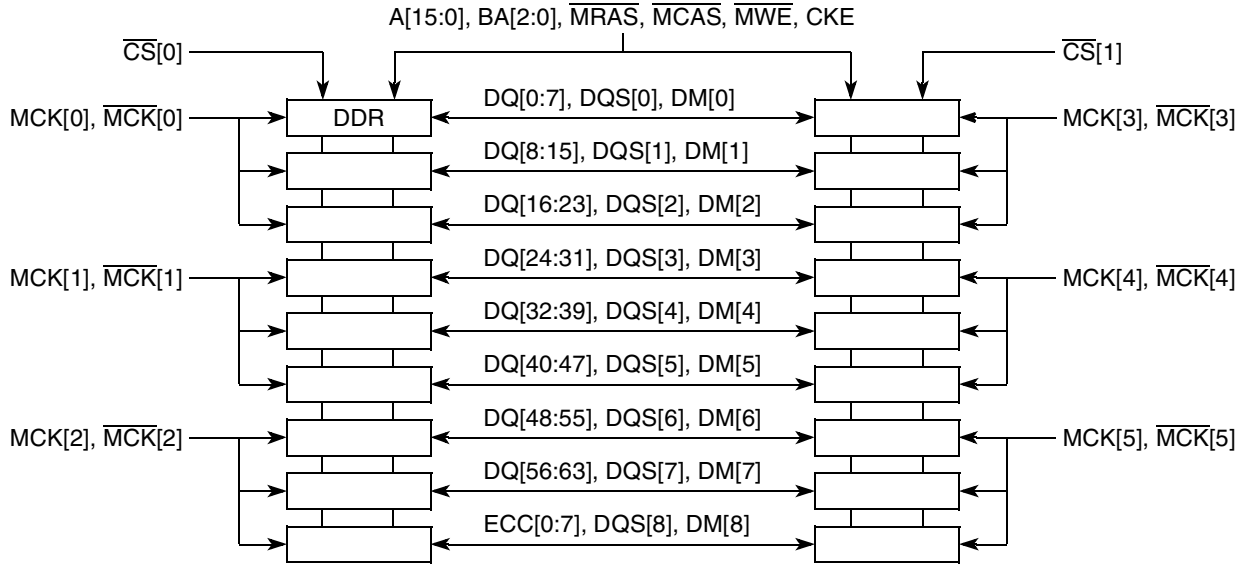


Figure 9-41. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

### 9.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the mode register set commands to the SDRAM array, and it uses the setting of TIMING\_CFG\_0[MRS\_CYC] for the Mode Register Set cycle time.

Figure 9-42 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

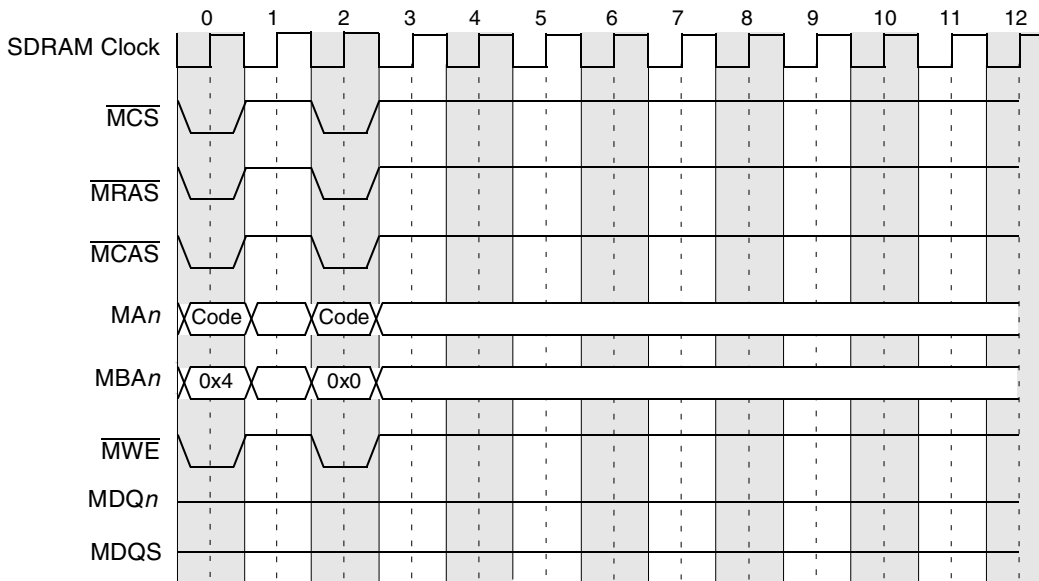


Figure 9-42. DDR SDRAM Mode-Set Command Timing



## 9.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting `DDR_SDRAM_CFG[RD_EN]` compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

Figure 9-43 shows the registered DDR SDRAM DIMM single-beat write timing.

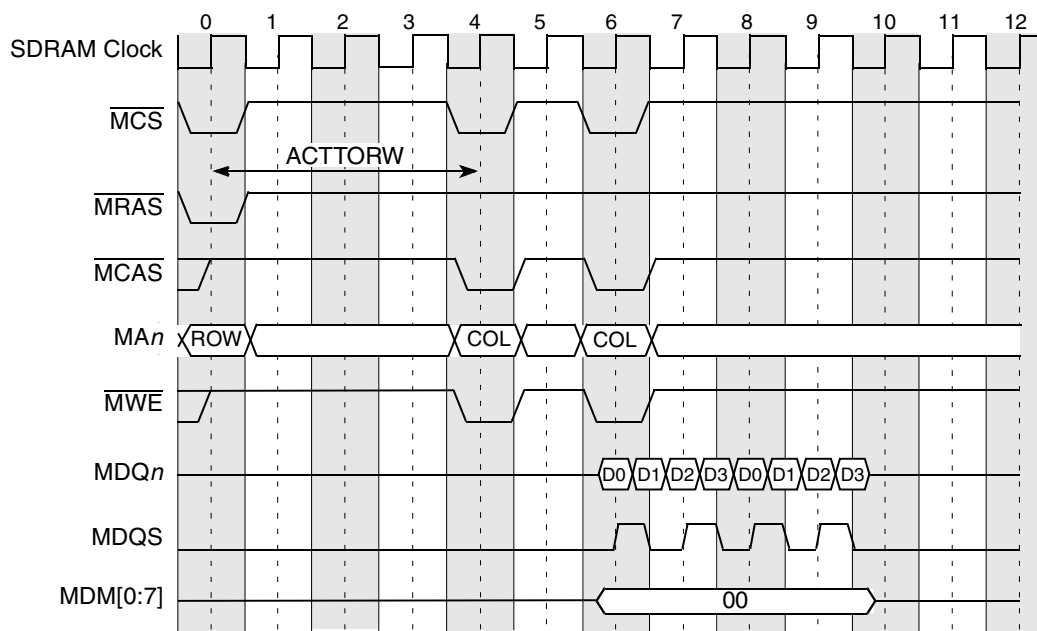


Figure 9-43. Registered DDR SDRAM DIMM Burst Write Timing

## 9.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (`TIMING_CFG_2[WR_DATA_DELAY]`) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The `WR_DATA_DELAY` parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMM. `TIMING_CFG_2[WR_DATA_DELAY]` specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

Figure 9-44 shows the use of the WR\_DATA\_DELAY parameter.

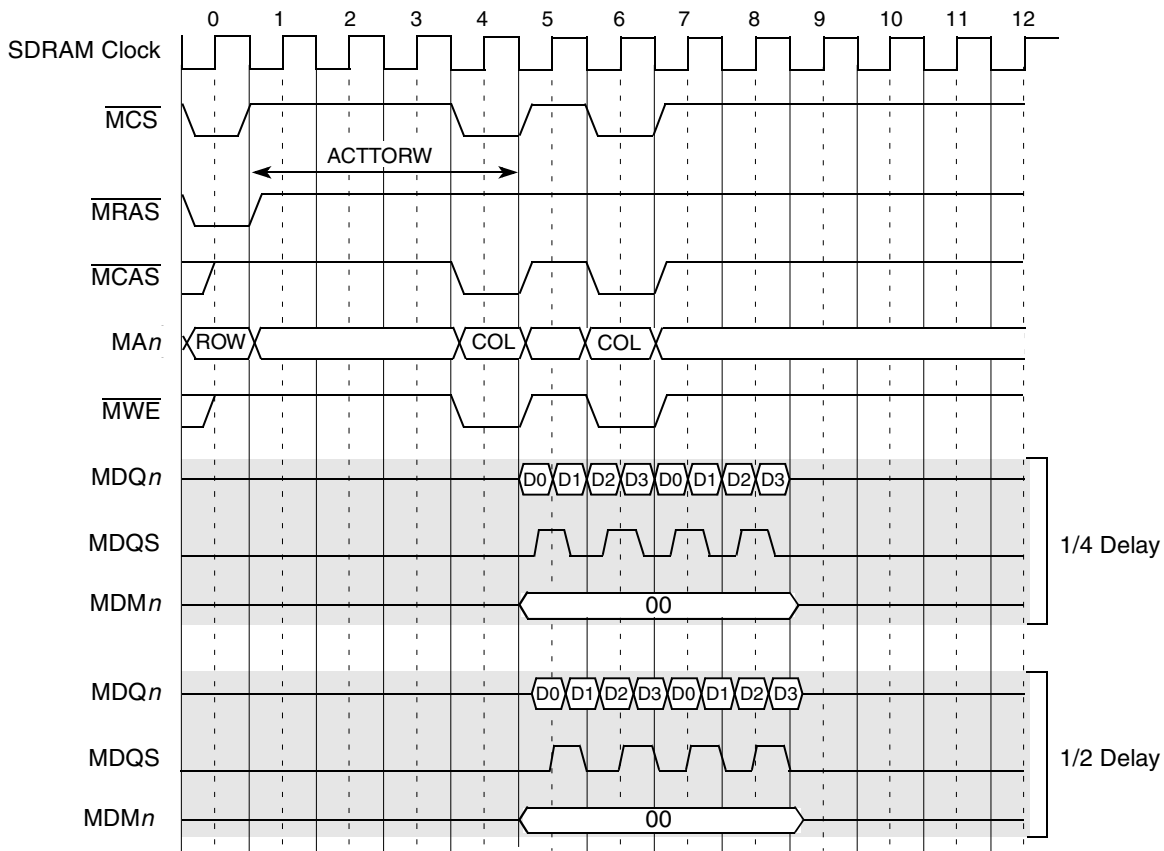


Figure 9-44. Write Timing Adjustments Example for Write Latency = 1

### 9.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the `DDR_SDRAM_INTERVAL[REFINT]` value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.
2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto-refresh commands are staggered across the four possible banks to reduce the system's instantaneous power requirements. Three sets of auto refresh commands will be issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in three groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. For this entire refresh sequence, no cycle optimization occurs for the usual case where fewer than four banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by `TIMING_CFG_1 [REFREC]` and `TIMING_CFG_3[EXT_REFREC]`. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

### 9.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter `TIMING_CFG_1 [REFREC]`, which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in [Figure 9-45](#) (`TIMING_CFG_1 [REFREC] = 10` in this example).

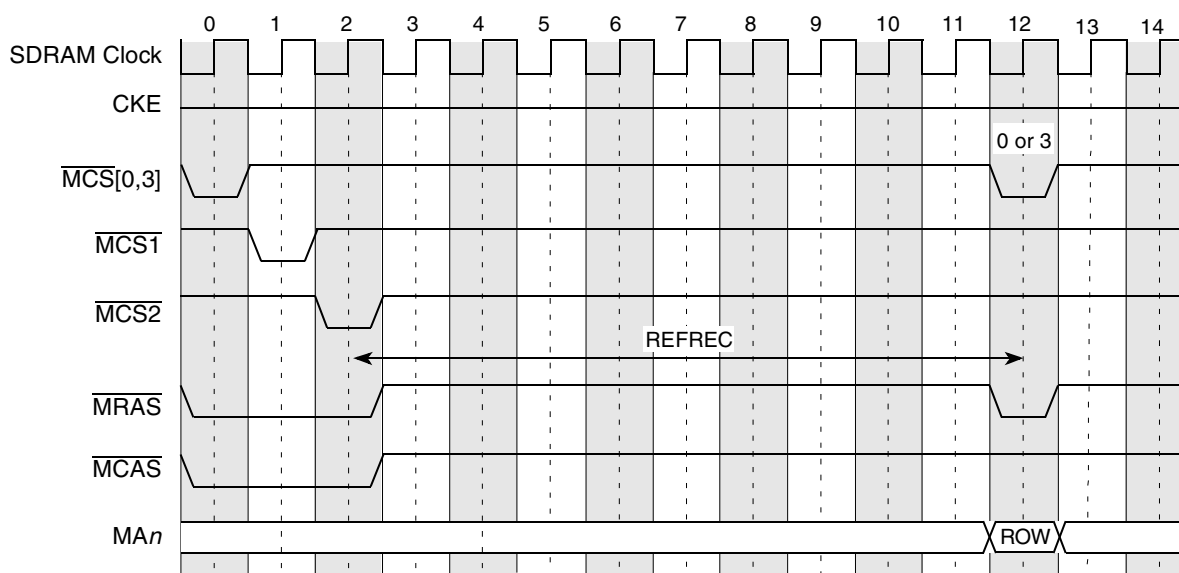


Figure 9-45. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of `TIMING_CFG_1 [REFREC]` and `TIMING_CFG_3[EXT_REFREC]` at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

### 9.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the `SREN` memory control parameter.

Table 9-48 summarizes the refresh types available in each power-saving mode.

**Table 9-48. DDR SDRAM Power-Saving Modes Refresh Configuration**

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

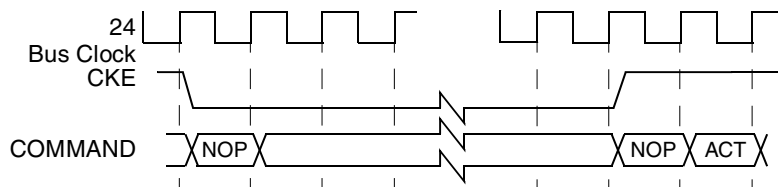
Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR\_SDRAM\_CFG[DYN\_PWR\_MGMT].

Dynamic power management mode offers tight control of the memory system’s power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING\_CFG\_0[ACT\_PD\_EXIT] and TIMING\_CFG\_0[PRE\_PD\_EXIT]. A penalty of 1 cycle is shown in Figure 9-46.



**Figure 9-46. DDR SDRAM Power-Down Mode**

### 9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-47 and Figure 9-48.

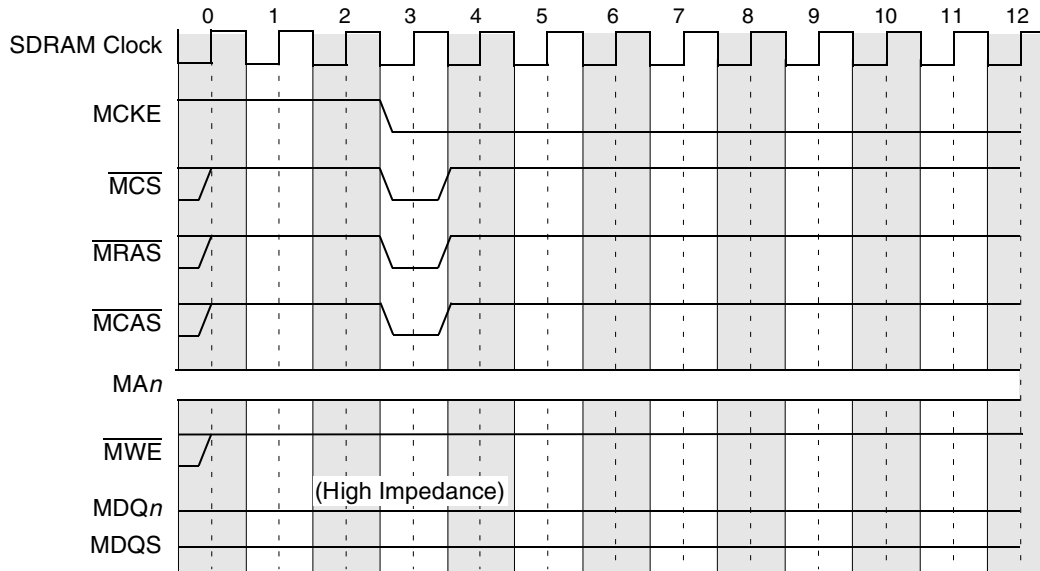


Figure 9-47. DDR SDRAM Self-Refresh Entry Timing

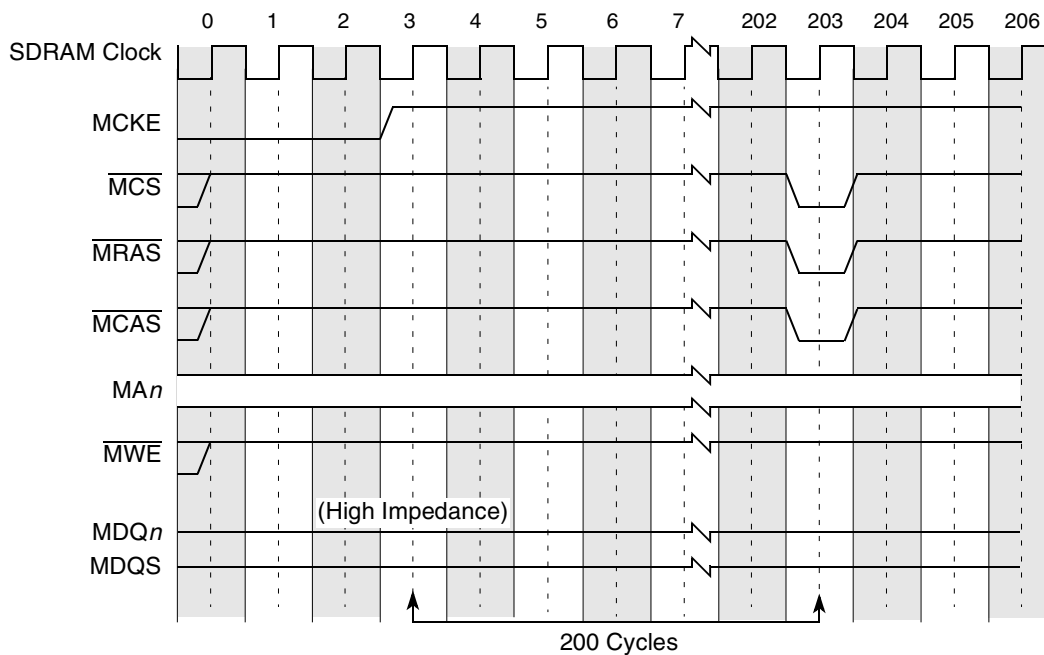


Figure 9-48. DDR SDRAM Self-Refresh Exit Timing

### 9.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not doubleword aligned or the size is not a multiple of a doubleword, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is doubleword aligned with a size that is a multiple of a doubleword, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the second, third, and fourth beats of data are not written to DRAM.

Table 9-49 lists the data beat sequencing to and from the DDR SDRAM and the data queues for each of the possible transfer sizes with each of the possible starting double-word offsets. All underlined double-word offsets are valid for the transaction.

**Table 9-49. Memory Controller–Data Beat Ordering**

Transfer Size	Starting Double-Word Offset	Double-Word Sequence <sup>1</sup> to/from DRAM and Queues
1 double word	0	<u>0</u> - 1 - 2 - 3
	1	<u>1</u> - 2 - 3 - 0
	2	<u>2</u> - 3 - 0 - 1
	3	<u>3</u> - 0 - 1 - 2
2 double words	0	<u>0</u> - <u>1</u> - 2 - 3
	1	<u>1</u> - <u>2</u> - 3 - 0
	2	<u>2</u> - <u>3</u> - 0 - 1
3 double words	0	<u>0</u> - <u>1</u> - <u>2</u> - 3
	1	<u>1</u> - <u>2</u> - <u>3</u> - 0

<sup>1</sup> All underlined **Double**-word offsets are valid for the transaction.

### 9.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains open until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR\_SDRAM\_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing `DDR_SDRAM_INTERVAL[BSTOPRE]` or setting `CSn_CONFIG[AP_nEN]`.

### 9.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in [Table 9-50](#) and [Table 9-51](#).

**Table 9-50. DDR SDRAM ECC Syndrome Encoding**

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	
9	•		•				•	
10	•			•			•	
11	•				•		•	
12	•	•				•	•	•
13	•		•			•	•	•
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	
41			•		•		•	
42	•		•		•		•	•
43		•	•		•		•	•
44			•	•		•	•	•
45			•		•	•	•	•

Table 9-50. DDR SDRAM ECC Syndrome Encoding (continued)

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•			•	•		
17		•		•				•	49			•		•	•		
18		•			•			•	50				•		•	•	
19	•	•			•				51	•				•	•		
20		•	•			•			52		•			•		•	
21		•		•		•			53			•		•		•	
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•				•		•	
24		•	•				•		56		•				•	•	
25		•		•				•	57			•			•	•	
26		•			•		•		58				•		•	•	
27	•	•			•		•	•	59	•					•	•	
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•



Table 9-51. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

### 9.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in [Section 9.4.1.27, “Memory Error Interrupt Enable \(ERR\\_INT\\_EN\),”](#) [Section 9.4.1.26, “Memory Error Disable \(ERR\\_DISABLE\),”](#) and [Section 9.4.1.25, “Memory Error Detect \(ERR\\_DETECT\).”](#)

Single-bit errors are counted and reported based on the ERR\_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR\_SBE[SBEC]
- Generates a critical interrupt if the counter value ERR\_SBE[SBEC] equals the programmable threshold ERR\_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the machine check or critical interrupt (if enabled, as described in [Section 9.4.1.26, “Memory Error Disable \(ERR\\_DISABLE\)”](#)). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in [Section 9.4.1.25, “Memory Error Detect \(ERR\\_DETECT\)”](#)). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. [Table 9-52](#) shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 9-52. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported via machine check or critical interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

## 9.6 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate  $\overline{MCS}_n$  signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in [Section 9.4.1.2, “Chip Select Configuration \(CS<sub>n</sub>\\_CONFIG\).”](#) Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See [Section 9.4.1, “Register Descriptions,”](#) for more detailed descriptions of the configuration registers. These parameters are shown in [Table 9-53.](#)

Table 9-53. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/page
CS <sub>n</sub> _BNDS	Chip select memory bounds	SAn EAn	<a href="#">9.4.1.1/9-11</a>
CS <sub>n</sub> _CONFIG	Chip select configuration	CS <sub>n</sub> _EN AP <sub>n</sub> _EN ODT_RD_CFG ODT_WR_CFG BA_BITS_CS <sub>n</sub> ROW_BITS_CS <sub>n</sub> COL_BITS_CS <sub>n</sub>	<a href="#">9.4.1.2/9-11</a>
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC	<a href="#">9.4.1.3/9-13</a>
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	<a href="#">9.4.1.4/9-14</a>
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD	<a href="#">9.4.1.5/9-16</a>
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE WR_DATA_DELAY CKE_PLS FOUR_ACT	<a href="#">9.4.1.6/9-18</a>

**Table 9-53. Memory Interface Configuration Register Initialization Parameters (continued)**

Name	Description	Parameter	Section/page	
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW	NCAP 2T_EN BA_INTLV_CTL x32_EN HSE BI	9.4.1.7/9-20
DDR_SDRAM_CFG_2	Control configuration	SR_IE DLL_RST_DIS DQS_CFG ODT_CFG	NUM_PR D_INIT	9.4.1.8/9-23
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE		9.4.1.9/9-25
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3		9.4.1.10/9-26
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE		9.4.1.12/9-29
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE		9.4.1.13/9-29
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST		9.4.1.14/9-30
DDR_INIT_ADDR	Initialization address	INIT_ADDR		9.4.1.15/9-30
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR		9.4.1.16/9-31

### 9.6.1 Programming Differences Between Memory Types

Depending on the memory type used, certain fields must be programmed differently. Table 9-54 illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

**Table 9-54. Programming Differences Between Memory Types**

Parameter	Description	Differences		Section/page
AP $n$ _EN	Chip Select $n$ Auto Precharge Enable	DDR1	Can be used to place chip select $n$ in auto precharge mode	9.4.1.2/9-11
		DDR2	Can be used to place chip select $n$ in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR1	Should always be set to 000	9.4.1.2/9-11
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.	

Table 9-54. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR1	Should always be set to 000	9.4.1.2/9-11
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR1	Should be set to 0000	9.4.1.4/9-14
		DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is $t_{AXPD}$ .	
PRETOACT	Precharge to Activate Timing	DDR1	Should be set according to the specifications for the memory used ( $t_{RP}$ )	9.4.1.5/9-16
		DDR2	Should be set according to the specifications for the memory used ( $t_{RP}$ )	
ACTTOPRE	Activate to Precharge Timing	DDR1	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used ( $t_{RAS}$ )	9.4.1.5/9-16
		DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used ( $t_{RAS}$ )	
ACTTORW	Activate to Read/Write Timing	DDR1	Should be set according to the specifications for the memory used ( $t_{RCD}$ )	9.4.1.5/9-16
		DDR2	Should be set according to the specifications for the memory used ( $t_{RCD}$ )	
CASLAT	CAS Latency	DDR1	Should be set, along with the Extended CAS Latency, to the desired CAS latency	9.4.1.5/9-16
		DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR1	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used ( $t_{RFC}$ )	9.4.1.5/9-16
		DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used ( $T_{RFC}$ )	
WRREC	Write Recovery	DDR1	Should be set according to the specifications for the memory used ( $t_{WR}$ )	9.4.1.5/9-16
		DDR2	Should be set according to the specifications for the memory used ( $t_{WR}$ )	
ACTTOACT	Activate <i>A</i> to Activate <i>B</i>	DDR1	Should be set according to the specifications for the memory used ( $t_{RRD}$ )	9.4.1.5/9-16
		DDR2	Should be set according to the specifications for the memory used ( $t_{RRD}$ )	

Table 9-54. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
WRTORD	Write to Read Timing	DDR1	Should be set according to the specifications for the memory used ( $t_{WRD}$ )	9.4.1.5/9-16
		DDR2	Should be set according to the specifications for the memory used ( $t_{WRD}$ )	
ADD_LAT	Additive Latency	DDR1	Should be set to 000	9.4.1.6/9-18
		DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Should be set to 001	9.4.1.6/9-18
		DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency is 5 cycles, then this field should be set to 100 (4 cycles).	
RD_TO_PRE	Read to Precharge Timing	DDR1	Should be set to 010 if burst length is 4 and 100 if burst length is 8	9.4.1.6/9-18
		DDR2	Should be set according to the specifications for the memory used ( $t_{RTP}$ ). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + $t_{RTP}$ cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001	9.4.1.6/9-18
		DDR2	Should be set according to the specifications for the memory used ( $t_{CKE}$ )	
FOUR_ACT	Four Activate Window	DDR1	Should be set to 0001	9.4.1.6/9-18
		DDR2	Should be set according to the specifications for the memory used ( $t_{FAW}$ ). Only applies to eight logical banks.	
RD_EN	Registered DIMM Enable	DDR1	If registered DIMMs are used, then this field should be set to 1	9.4.1.7/9-20
		DDR2	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR1	If a 32-bit bus is used, and 8-beat bursts are desired, then this field should be set to 1	9.4.1.7/9-20
		DDR2	Should be set to 0	
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	9.4.1.7/9-20
		DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	

Table 9-54. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
DLL_RST_DIS	DLL Reset Disable	DDR1	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	9.4.1.8/9-23
		DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	
DQS_CFG	DQS Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Can be set to either 00 or 01, depending on if differential strobes will be used	
ODT_CFG	ODT Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
BSTOPRE	Burst To Precharge Interval	DDR1	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	9.4.1.12/9-29
		DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

## 9.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set `DDR_SDRAM_CFG[MEM_EN]` to enable the memory interface. Note that 200  $\mu$ s must elapse after DRAM clocks are stable (`DDR_SDRAM_CLK_CNTL[CLK_ADJUST]` is set and any chip select is enabled) before `MEM_EN` can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If `DDR_SDRAM_CFG[BI]` is not set, the DDR memory controller will conduct an automatic initialization sequence to the memory, which will follow the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the `DDR_SDRAM_MD_CNTL` register.

## 9.6.3 Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System

This section describes the options offered by this device to support battery-backed main memory.

### 9.6.3.1 Hardware Based Self-Refresh

An external voltage sense device can be connected to this device through one of the external interrupt lines `IRQn`. The external interrupt from the voltage sensor would then be steered through this device's programmable interrupt controller (PIC) to the `IRQ_OUT` signal. Note that the `IRQ_OUT` signal must remain high until power is removed.

If `DDR_SDRAM_CFG_2[SR_IE]` is set, the `IRQ_OUT` signal from the interrupt controller is then automatically detected by the DDR controller, which immediately causes main memory to enter self-refresh mode. See [Section 9.4.1.8, “DDR SDRAM Control Configuration 2 \(DDR\\_SDRAM\\_CFG\\_2\),”](#) for further information on this bit.

These fields in the appropriate registers in the PIC must be set for self refresh to function:

- `EIVPRn[PRIORITY]` should be set to 0xF (highest priority)
- `EIDRn[EP]` should be set in order to route the incoming signal to `IRQ_OUT`

See [Section 10.3.7.1, “External Interrupt Vector/Priority Registers \(EIVPR0–EIVPR11\),”](#) and [Section 10.3.7.2, “External Interrupt Destination Registers \(EIDR0–EIDR11\),”](#) for descriptions of these registers.

Note that this application precludes any other simultaneous use of `IRQ_OUT`.

### 9.6.3.2 Software Based Self-Refresh

The DDR controller also has a software-programmable bit, `DDR_SDRAM_CFG_2[FRC_SR]`, that immediately puts main memory into self-refresh mode. See [Section 9.4.1.8, “DDR SDRAM Control Configuration 2 \(DDR\\_SDRAM\\_CFG\\_2\),”](#) for a description of this register.

It is expected that a critical interrupt routine triggered by an external voltage sensing device will have time to set this bit.

### 9.6.3.3 Bypassing Re-initialization During Battery-Backed Operation

The DDR controller offers an initialization bypass feature (`DDR_SDRAM_CFG[BI]`), which system designers may use to prevent re-initialization of main memory during system power-on following an abnormal shutdown. See [Section 9.4.1.7, “DDR SDRAM Control Configuration \(DDR\\_SDRAM\\_CFG\),”](#) for information on this bit and [Section 9.4.1.15, “DDR Initialization Address \(DDR\\_INIT\\_ADDR\),”](#) for a discussion of avoiding possible ECC errors in this mode.

Note that the DDR controller will automatically wait 200 DRAM cycles before issuing any command after the assertion of `MCKE[0:3]` when this mode is used.





---

# Chapter 10

## Programmable Interrupt Controller

This chapter describes the programmable interrupt controller (PIC) interrupt protocol, various types of interrupt sources controlled by the PIC unit, and the PIC registers with some programming guidelines.

### 10.1 Introduction

The PIC prioritizes and manages interrupts directed to the  $\overline{int}$  signal. It also manages the interrupts generated by the PIC itself and by off-chip interrupt sources.

The PIC receives interrupt signals from three sources: external to the integrated device, internal to the integrated device, and intrinsic to the PIC itself. The PIC selects among all current interrupts the one with the highest priority and forwards it to the internal processor core, or, in pass-through mode, off-chip for external servicing.

#### 10.1.1 Overview

The PIC is compliant with the OpenPIC architecture. The interrupt controller provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.

A high level block diagram of the peripheral interrupt controller (PIC) showing the interfaces is shown in Figure 10-1.

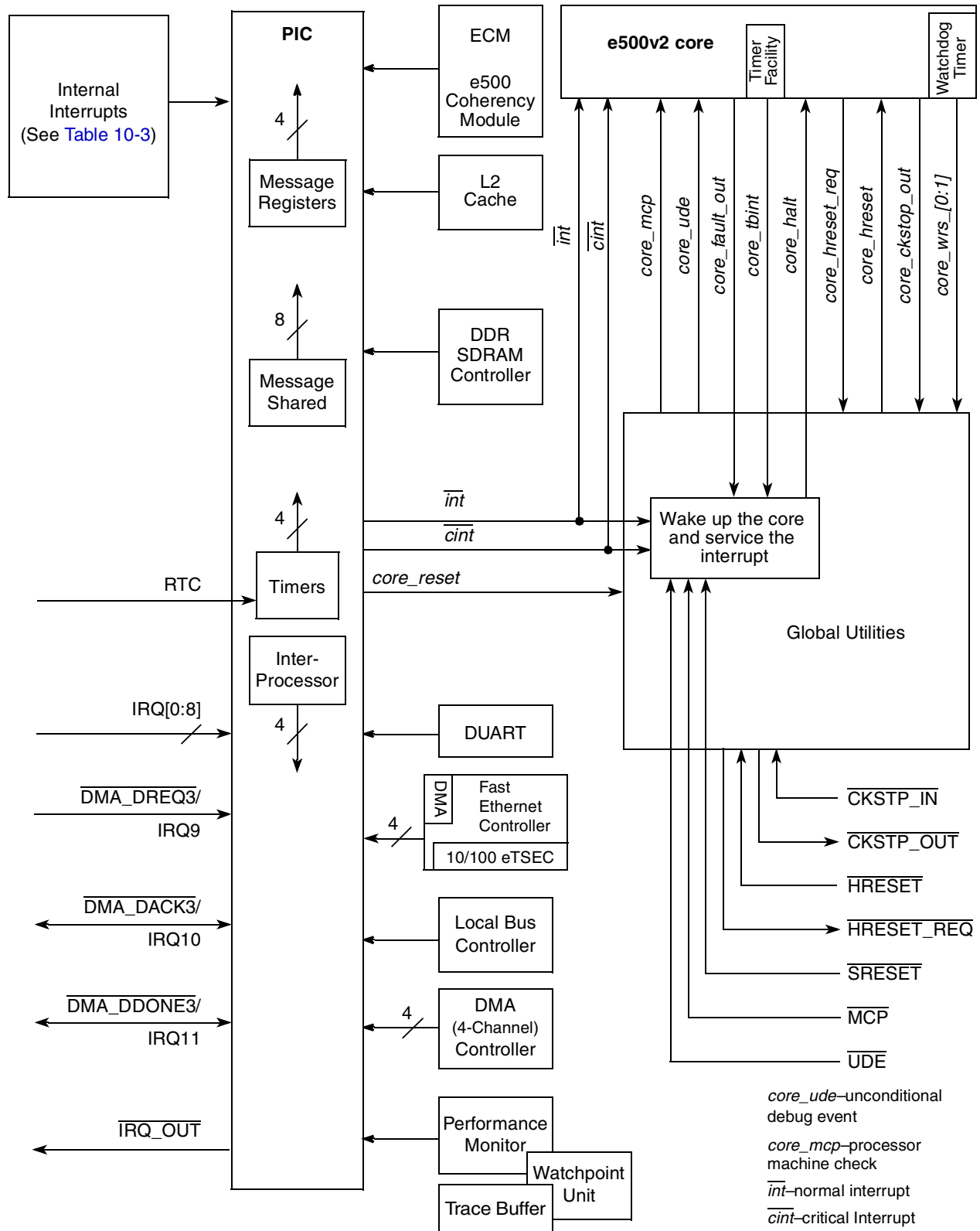


Figure 10-1. Interrupt Sources Block Diagram

## 10.1.2 Features

- Programming model compliant with the OpenPIC architecture
- Support for 12 external and 48 internal interrupt sources. (Table 10-5 lists which of the 48 internal sources are implemented.) Serial interrupts are not supported.
- Four interprocessor interrupt channels.
- Four 32-bit messaging interrupt channels.
- Eight shared message signaled interrupt sources and up to 32 sharers for shared interrupt register.
- Four global high-resolution timers that can be clocked with the platform clock or the RTC input.
- Fully-nested interrupt delivery
- Processor initialization control
- Programmable resetting of the PIC unit through the global configuration register
- 16 programmable interrupt priority levels
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller
- In 8259 mode, it generates a local (internal) interrupt output signal,  $\overline{\text{IRQ\_OUT}}$ .
- Recovery from spurious interrupts

## 10.1.3 Interrupts to the Processor Core

The  $\overline{\text{int}}$  signal, which causes the external interrupt exception, is the main interrupt output from the PIC unit to the processor core. Interrupts can alternately be configured as critical interrupts (in the destination registers); these are reported to the core through the  $\overline{\text{cint}}$  signal. The architecture implemented by the e500 core defines a separate critical interrupt type with its own save and restore registers (CSRR0 and CSRR1) and return instruction (Return from Critical Interrupt, **rfdi**). In addition to the external and critical interrupts that are generated by the PIC, other conditions (shown in Table 10-1) cause interrupts to the core (and wake up the core when it is in a low-power state).

**Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources**

Core Interrupt Type	Signaled by (Input to Core)	Sources
<b>PIC—Programmable Interrupts</b>		
External interrupt	$\overline{\text{int}}$	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
Critical interrupt	$\overline{\text{cint}}$	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
<b>Other Interrupts Generated Outside the Core</b>		
Machine check	<i>core_mcp</i>	<ul style="list-style-type: none"> <li>• <math>\overline{\text{MCP}}</math></li> <li>• <math>\overline{\text{SRESET}}</math></li> <li>• Assertion of <i>core_mcp</i> by global utilities block</li> </ul>

**Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources (continued)**

Core Interrupt Type	Signaled by (Input to Core)	Sources
Unconditional debug event	<i>core_ude</i>	$\overline{UDE}$ . Asserting $\overline{UDE}$ generates an unconditional debug exception type debug interrupt and sets a bit in the debug status register, DBSR[UDE], as described in <a href="#">Section 6.13.2, “Debug Status Register (DBSR).”</a>
Reset	<i>core_hreset</i>	<ul style="list-style-type: none"> <li><math>\overline{HRESET}</math> assertion (and negation)</li> <li><i>core_hreset_req</i>. Output from core—caused by writing to the core DBCR0[RST]. This condition is additionally qualified with MSR[DE] and DBCR0[IDM] bits. Note that assertion of this signal causes a hard reset of the core only. <i>core_hreset_req</i> can also be caused by a second timer timeout condition as described in <a href="#">Section 10.3.2.6, “Timer Control Register (TCR).”</a></li> <li><i>core_reset</i>. Output from PIC. See <a href="#">Section 10.3.1.6, “Processor Initialization Register (PIR).”</a></li> </ul>

The global utilities block monitors two additional interrupt conditions generated by the e500 core (*core\_tshint* and *core\_fault\_out* signals), as shown in [Table 10-2](#). Assertion of either of these signals causes the processor to exit a low-power state. These cases are caused by core conditions, and after the global utilities logic wakes up the core, they are handled by the core as shown in [Table 10-2](#).

**Table 10-2. e500 Core-Generated Interrupts that Cause a Wake-Up**

Core Interrupt Type	Signaled by (Output from Core)	Sources
Fixed interval timer	<i>core_tshint</i>	The source of both of these interrupts is the time base facility within the e500 core. The integrated logic monitors this core output signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state). For more information about the interaction between core-generated signals and power management, see <a href="#">Chapter 19, “Global Utilities.”</a>
Decrementer		
Machine check	<i>core_fault_out</i>	Occurs when the L1 cache has a parity error on a snoop push operation, which can occur while the core is halted. The integrated logic monitors this signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state).

### 10.1.4 Modes of Operation

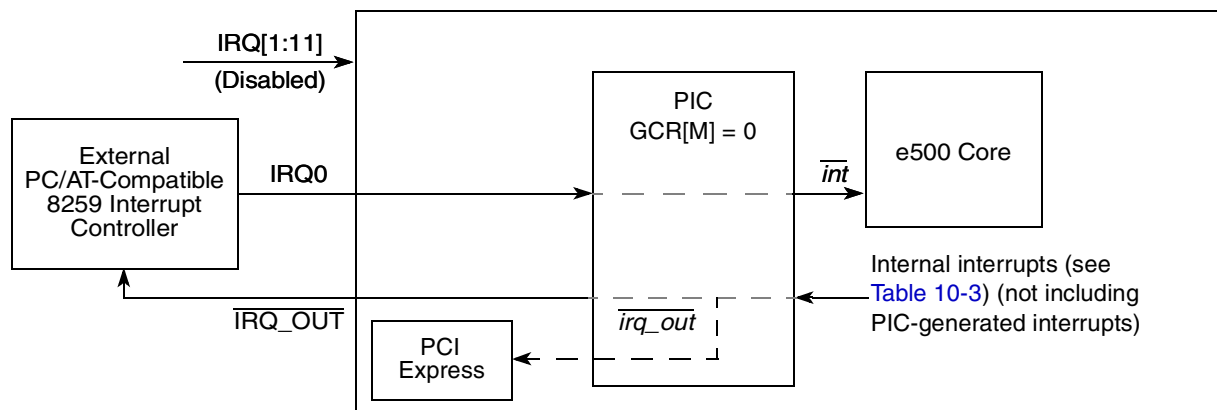
Mixed or pass-through mode can be chosen by setting or clearing GCR[M] as described in [Section 10.3.1.4, “Global Configuration Register \(GCR\).”](#)

#### 10.1.4.1 Mixed Mode (GCR[M] = 1)

In mixed mode, the external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in [Section 10.3.7.1, “External Interrupt Vector/Priority Registers \(EIVPR0–EIVPR11\),”](#) through [Section 10.3.7.4, “Internal Interrupt Destination Registers \(IIDR0–IIDR47\).”](#)

### 10.1.4.2 Pass-Through Mode (GCR[M] = 0)

The PIC unit provides a mechanism to support alternate external interrupt controllers such as the PC/AT compatible 8259 interrupt controller architecture. After a hard reset, the PIC unit defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to the e500 core, as shown in [Table 10-2](#); all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor. The PIC does not perform a vector fetch from an 8259 interrupt controller.



**Figure 10-2. Pass-Through Mode Example**

When pass-through mode is enabled, the internally-generated interrupts shown in [Table 10-5](#), are not forwarded to the e500 core. Instead, the PIC passes the raw interrupts from the internal sources to IRQ\_OUT.

Note that when the PCI Express controller is configured as an endpoint (EP) device, the *irq\_out* signal from the PIC may be used to automatically generate an outbound PCI Express MSI transaction toward the remote interrupt controller resource on the root complex (RC). See [Section 18.4.2.1.2, “Hardware MSI Generation.”](#)

Note that in pass-through mode, interrupts generated by the PIC itself (global timers, interprocessor, and message register interrupts) cannot be used. If internal or PIC-generated interrupts must be reported internally to the processor, pass-through mode must be disabled.

## 10.1.5 Interrupt Sources

Aside from the sources of machine check, unconditional debug event, and reset interrupts to the core described in [Table 10-1](#), the PIC unit can receive interrupts from six different sources as follows:

- External—Off-chip signals, IRQ[0:11]
- Internal—On-chip. Sources are L2, ECM, DDR, LBC, DMA, PCI Express, eTSEC1–2, DUART, performance monitor, security engine, and I<sup>2</sup>C.
- Global timers—From inside the PIC
- Inter-processor (IPI)—Intended for communication between different processor cores on the same device. Used only for self-interrupt in single-core devices.

- Message registers—From inside the PIC. Triggered on register write, cleared on read. Used for inter-process communication.
- Shared message signaled interrupt registers—From within the PIC. Triggered on register write; cleared on read.

### 10.1.5.1 Interrupt Routing—Mixed Mode

When an interrupt request is delivered to the PIC, the corresponding interrupt destination register is checked to determine where the request should be routed, as follows:

- If  $xIDRn[EP] = 1$  (and all other destination bits are zero), the interrupt is routed off-chip to the external signal. Or if the PCI Express controller is in EP mode and automatically generates a PCI Express MSI transaction. See [Section 18.4.2.1.2, “Hardware MSI Generation.”](#)
- If  $xIDRn[CI]$  is set (and all other destination bits are zero), the interrupt is routed to  $\overline{cint}$ .
- If  $xIDRn[P0]$  is set (and all other destination bits are zero) the interrupt is routed to  $int0$ . Setting  $xIDRn[P1]$  likewise routs the interrupt to  $int1$ . In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow is as described in [Section 10.4.1, “Flow of Interrupt Control.”](#)

### 10.1.5.2 Internal Interrupt Sources

[Table 10-3](#) shows the assignments of the internal interrupt sources. Note that this list does not include the interrupts generated by the PIC unit.

**Table 10-3. Internal Interrupt Sources<sup>1</sup>**

Internal Interrupt Number	Interrupt Source	Internal Interrupt Number	Interrupt Source
0	L2 cache	24	Reserved
1	ECM	25	Reserved
2	DDR DRAM	26	DUART
3	LBC	27	I <sup>2</sup> C controllers
4	DMA channel 0	28	Performance monitor
5	DMA channel 1	29	Security channel
6	DMA channel 2	30	Reserved
7	DMA channel 3	31	Reserved
8	PCI	32	Reserved
9	PCI Express 2	33	Reserved
10	PCI Express1	34	Reserved
11	PCI Express 3	35	Reserved
12	Reserved	36	Reserved
13	eTSEC1 transmit	37	Reserved

**Table 10-3. Internal Interrupt Sources<sup>1</sup> (continued)**

Internal Interrupt Number	Interrupt Source	Internal Interrupt Number	Interrupt Source
14	eTSEC1 receive	38	Reserved
15	eTSEC3 transmit	39	Reserved
16	eTSEC3 receive	40	Reserved
17	eTSEC3 error	41	Reserved
18	eTSEC1 error	42	Reserved
19	Reserved	43	Reserved
20	Reserved	44	Reserved
21	Reserved	45	Reserved
22	Reserved	46	Reserved
23	Reserved	47	Reserved

<sup>1</sup> The general intent of these Internal Interrupt Sources is to maintain consistency across PowerQUICC III family devices.

## 10.2 External Signal Descriptions

The following sections provide an overview and detailed descriptions of the PIC signals.

### 10.2.1 Signal Overview

PIC interface signals are described in [Table 10-4](#). There are 12 distinct external interrupt request input signals (IRQ[0:11]) and 1 interrupt request output signal ( $\overline{\text{IRQ\_OUT}}$ ).

**Table 10-4. PIC Interface Signals**

Signal Name	I/O	Description
IRQ[0:11]	I	External interrupts
$\overline{\text{IRQ\_OUT}}$	O	Interrupt request out
$\overline{\text{MCP}}$	I	Processor machine check
$\overline{\text{UDE}}$	I	Unconditional debug event

## 10.2.2 Detailed Signal Descriptions

Table 10-5 provides detailed descriptions of the external PIC signals.

**Table 10-5. Interrupt Signals—Detailed Signal Descriptions**

Signal	I/O	Description
IRQ[0:11]	I	Interrupt request 0–11. The polarity and sense of each of these signals is programmable. All of these inputs can be driven asynchronously. The interrupt request signals IRQn may share PIC external interrupt registers with PCI Express INTx signaling. See <a href="#">Section 10.4.6, “PCI Express INTx,”</a> for more information.
		<p><b>State Meaning</b></p> <p>Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the priority is checked by the PIC unit, and the interrupt is conditionally passed to the processor. In pass-through mode, only interrupts detected on IRQ0 are passed directly to the processor core.</p> <p>Negated—There is no incoming interrupt from that source.</p>
		<p><b>Timing</b></p> <p>Assertion—All of these inputs can be asserted completely asynchronously.</p> <p>Negation—Interrupts programmed as level-sensitive must remain asserted until serviced.</p>
$\overline{\text{IRQ\_OUT}}$	O	Interrupt request out. Active-low, open drain. When the PIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See <a href="#">Section 10.1.4, “Modes of Operation,”</a> for more details.
		<p><b>State Meaning</b></p> <p>Asserted—At least one interrupt is currently being signaled to the external system.</p> <p>Negated—Indicates no interrupt source currently routed to <math>\overline{\text{IRQ\_OUT}}</math>.</p>
		<p><b>Timing</b></p> <p>Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of <math>\overline{\text{IRQ\_OUT}}</math> occurs asynchronously with respect to the interrupt source. All timing given here is approximate.</p> <p>Assertion—Internal interrupt source: 2 CCB clock cycles after interrupt occurs.                      External interrupt source: 4 cycles after interrupt occurs.                      Message interrupts: 2 cycles after write to message register.</p> <p>Negation—Follows interrupt source negation with the following delay:                      Internal interrupt: 2 CCB clock cycles                      External interrupt: 4 cycles.                      Message interrupts: 2 cycles after message register cleared.</p>
$\overline{\text{MCP}}$	I	Machine check processor. Assertion causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts ( $\text{MSR}[\text{ME}] = 0$ ), assertion of $\overline{\text{MCP}}$ causes a checkstop condition. Note that internal sources for the internal <i>core_mcp</i> signal can also cause a machine check interrupt to the processor core, as described in <a href="#">Section 19.4.1.14, “Machine Check Summary Register (MCPSUMR),”</a> <a href="#">Table 10-1</a> and <a href="#">Table 10-2</a> .
		<p><b>State Meaning</b></p> <p>Asserted—Integrated logic should initiate a machine check interrupt or enter the checkstop state as directed by the MSR.</p> <p>Negated—Machine check handling is not being requested by the external system.</p>
		<p><b>Timing</b></p> <p>Assertion—May occur at any time, asynchronous to any clock.</p> <p>Negation—Because <math>\overline{\text{MCP}}</math> is edge-triggered, it can be negated one clock after its assertion.</p>
$\overline{\text{UDE}}$	I	Unconditional debug event. Assertion signal causes an unconditional debug exception to the e500 core.
		<p><b>State Meaning</b></p> <p>Asserted—Indicates that integrated logic should initiate an unconditional debug event interrupt to the processor core.</p> <p>Negated—Indicates that unconditional debug event handling is not being requested by <math>\overline{\text{UDE}}</math>.</p>
		<p><b>Timing</b></p> <p>Assertion—May occur at any time, asynchronous to any clock.</p> <p>Negation—Should remain asserted until software in the unconditional debug event interrupt handler causes the external device asserting the <math>\overline{\text{UDE}}</math> signal to negate it.</p>



## 10.3 Memory Map/Register Definition

The PIC programmable register map occupies 256 Kbytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All PIC registers are 32 bits wide and, although located on 128-bit address boundaries, should only be accessed as 32-bit quantities.

The PIC address offset map, shown in [Table 10-6](#), is divided into three areas:

- 0xnn4\_0000–0xnn4\_FFF0—Global registers
- 0xnn5\_0000–0xnn5\_FFF0—Interrupt source configuration registers
- 0xnn6\_0000–0xnn7\_FFF0—Per-CPU registers

In the following table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 10-6. PIC Register Address Map**

Offset	Register	Access	Reset	Section/Page
<b>PIC Register Address Map—Global Registers</b>				
0x4_0000	BRR1—Block revision register 1	R	0x0040_0200	<a href="#">10.3.1.1/10-18</a>
0x4_0010	BRR2—Block revision register 2	R	0x0000_0001	<a href="#">10.3.1.2/10-19</a>
0x4_0020– 0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	<a href="#">10.3.8.1/10-45</a>
0x4_0050	IPIDR1—Interprocessor interrupt 1 (IPI 1) dispatch register			
0x4_0060	IPIDR2—Interprocessor interrupt 2 (IPI 2) dispatch register			
0x4_0070	IPIDR3—Interprocessor interrupt 3 (IPI 3) dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	<a href="#">10.3.8.2/10-46</a>
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	<a href="#">10.3.8.3/10-47</a>
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	<a href="#">10.3.8.4/10-47</a>
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	<a href="#">10.3.8.5/10-48</a>
0x4_00C0– 0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x004F_0002	<a href="#">10.3.1.3/10-19</a>

**Table 10-6. PIC Register Address Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.4/10-20
0x4_1030	Reserved	—	—	—
0x4_1040– 0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.5/10-21
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.6/10-21
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.7/10-22
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.8/10-23
<b>Global Timer Registers</b>				
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-23
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1200– 0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-26

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
<b>External, <math>\overline{\text{IRQ\_OUT}}</math>, and Critical Interrupt Summary Registers</b>				
0x4_1308	ERQSR—External interrupt summary register	R	0x0000_0000	10.3.3.1/10-28
0x4_1310	IRQSR0— $\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.2/10-29
0x4_1320	IRQSR1— $\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.3/10-30
0x4_1324	IRQSR2— $\overline{\text{IRQ\_OUT}}$ summary register 2	R	0x0000_0000	10.3.3.4/10-30
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.5/10-31
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.6/10-31
0x4_1344	CISR2—Critical Interrupt summary register 2	R	0x0000_0000	10.3.3.7/10-32
<b>Performance Monitor Mask Registers</b>				
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_13D0– 0x4_13F0	Reserved	—	—	—
<b>Message Registers</b>				
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-34
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-35
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-35

**Table 10-6. PIC Register Address Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x4_1520– 0x4_15F0	Reserved	—	—	—
0x4_1600	MSIR0—Shared message signaled interrupt register 0	Special	0x0000_0000	10.3.6.1/10-36
0x4_1610	MSIR1—Shared message signaled interrupt register 1	Special	0x0000_0000	10.3.6.1/10-36
0x4_1620	MSIR2—Shared message signaled interrupt register 2	Special	0x0000_0000	10.3.6.1/10-36
0x4_1630	MSIR3—Shared message signaled interrupt register 3	Special	0x0000_0000	10.3.6.1/10-36
0x4_1640	MSIR4—Shared message signaled interrupt register 4	Special	0x0000_0000	10.3.6.1/10-36
0x4_1650	MSIR5—Shared message signaled interrupt register 5	Special	0x0000_0000	10.3.6.1/10-36
0x4_1660	MSIR6—Shared message signaled interrupt register 6	Special	0x0000_0000	10.3.6.1/10-36
0x4_1670	MSIR7—Shared message signaled interrupt register 7	Special	0x0000_0000	10.3.6.1/10-36
0x4_1680– 0x4_1700	Reserved	—	—	—
0x4_1720	MSISR—Shared message signaled interrupt status register	R	0x0000_0000	10.3.6.2/10-36
0x4_1740	MSIIR—Shared message signaled interrupt index register	W	0x0000_0000	10.3.6.3/10-37
0x4_1750– 0x4_FFFF	Reserved	—	—	—
<b>PIC Register Address Map—Interrupt Source Configuration Registers</b>				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	Mixed	0x8000_0000	10.3.7.1/10-39
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	Mixed	0x0000_0001	10.3.7.2/10-40
0x5_0180– 0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0210	IIDR0—Internal interrupt 0 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0230	IIDR1—Internal interrupt 1 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0250	IIDR2—Internal interrupt 2 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0270	IIDR3—Internal interrupt 3 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0290	IIDR4—Internal interrupt 4 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02B0	IIDR5—Internal interrupt 5 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02D0	IIDR6—Internal interrupt 6 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02F0	IIDR7—Internal interrupt 7 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0310	IIDR8—Internal interrupt 8 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0330	IIDR9—Internal interrupt 9 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0350	IIDR10—Internal interrupt 10 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0370	IIDR11—Internal interrupt 11 destination register	Mixed	0x0000_0001	10.3.7.4/10-42

**Table 10-6. PIC Register Address Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0390	IIDR12—Internal interrupt 12 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03B0	IIDR13—Internal interrupt 13 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03D0	IIDR14—Internal interrupt 14 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03F0	IIDR15—Internal interrupt 15 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0410	IIDR16—Internal interrupt 16 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0430	IIDR17—Internal interrupt 17 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0450	IIDR18—Internal interrupt 18 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0470	IIDR19—Internal interrupt 19 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0490	IIDR20—Internal interrupt 20 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04B0	IIDR21—Internal interrupt 21 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04D0	IIDR22—Internal interrupt 22 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04F0	IIDR23—Internal interrupt 23 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0510	IIDR24—Internal interrupt 24 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0530	IIDR25—Internal interrupt 25 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0550	IIDR26—Internal interrupt 26 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0570	IIDR27—Internal interrupt 27 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0590	IIDR28—Internal interrupt 28 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05B0	IIDR29—Internal interrupt 29 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05D0	IIDR30—Internal interrupt 30 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05F0	IIDR31—Internal interrupt 31 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0600	IIVPR32—Internal interrupt 32 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0610	IIDR32—Internal interrupt 32 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0620	IIVPR33—Internal interrupt 33 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0630	IIDR33—Internal interrupt 33 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0640	IIVPR34—Internal interrupt 34 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0650	IIDR34—Internal interrupt 34 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0660	IIVPR35—Internal interrupt 35 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0670	IIDR35—Internal interrupt 35 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0680	IIVPR36—Internal interrupt 36 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0690	IIDR36—Internal interrupt 36 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_06A0	IIVPR37—Internal interrupt 37 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_06B0	IIDR37—Internal interrupt 37 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_06C0	IIVPR38—Internal interrupt 38 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_06D0	IIDR38—Internal interrupt 38 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_06E0	IIVPR39—Internal interrupt 39 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_06F0	IIDR39—Internal interrupt 39 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0700	IIVPR40—Internal interrupt 40 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0710	IIDR40—Internal interrupt 40 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0720	IIVPR41—Internal interrupt 41 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0730	IIDR41—Internal interrupt 41 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0740	IIVPR42—Internal interrupt 42 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0750	IIDR42—Internal interrupt 42 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0760	IIVPR43—Internal interrupt 43 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0770	IIDR43—Internal interrupt 43 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0780	IIVPR44—Internal interrupt 44 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0790	IIDR44—Internal interrupt 44 destination register	Mixed	0x0000_0001	10.3.7.4/10-42



**Table 10-6. PIC Register Address Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x5_07A0	IIVPR45—Internal interrupt 45 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_07B0	IIDR45—Internal interrupt 45 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_07C0	IIVPR46—Internal interrupt 46 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_07D0	IIDR46—Internal interrupt 46 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_07E0	IIVPR47—Internal interrupt 47 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_07F0	IIDR47—Internal interrupt 47 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0800– 0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	Mixed	0x8000_0000	10.3.7.5/10-43
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	Mixed	0x0000_0001	10.3.7.6/10-43
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	Mixed	0x8000_0000	10.3.7.5/10-43
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	Mixed	0x0000_0001	10.3.7.6/10-43
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	Mixed	0x8000_0000	10.3.7.5/10-43
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	Mixed	0x0000_0001	10.3.7.6/10-43
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	Mixed	0x8000_0000	10.3.7.5/10-43
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	Mixed	0x0000_0001	10.3.7.6/10-43
0x5_1680– 0x5_1BF0	Reserved	—	—	—
0x5_1C00	MSIVPR0—Shared message signaled interrupt vector/priority register 0	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C10	MSIDR0—Shared message signaled interrupt destination register 0	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C20	MSIVPR1—Shared message signaled interrupt vector/priority register 1	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C30	MSIDR1—Shared message signaled interrupt destination register 1	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C40	MSIVPR2—Shared message signaled interrupt vector/priority register 2	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C50	MSIDR2—Shared message signaled interrupt destination register 2	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C60	MSIVPR3—Shared message signaled interrupt vector/priority register 3	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C70	MSIDR3—Shared message signaled interrupt destination register 3	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C80	MSIVPR4—Shared message signaled interrupt vector/priority register 4	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C90	MSIDR4—Shared message signaled interrupt destination register 4	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CA0	MSIVPR5—Shared message signaled interrupt vector/priority register 5	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1CB0	MSIDR5—Shared message signaled interrupt destination register 5	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CC0	MSIVPR6—Shared message signaled interrupt vector/priority register 6	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1CD0	MSIDR6—Shared message signaled interrupt destination register 6	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CE0	MSIVPR7—Shared message signaled interrupt vector/priority register 7	Mixed	0x1000_0000	10.3.6.4/10-38



Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_1CF0	MSIDR7—Shared message signaled interrupt destination register 7	Mixed	0x0000_0001	<a href="#">10.3.6.5/10-38</a>
0x5_1D00– 0x5_FFF0	Reserved	—	—	—
<b>PIC Register Address Map—Per-CPU Registers</b>				
0x6_0000– 0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	<a href="#">10.3.8.1/10-45</a>
0x6_0050	IPIDR1—Interprocessor interrupt 1 (IPI 1) dispatch register			
0x6_0060	IPIDR2—Interprocessor interrupt 2 (IPI 2) dispatch register			
0x6_0070	IPIDR3—Interprocessor interrupt 3 (IPI 3) dispatch register			
0x6_0080	CTPR—Current task priority register	R/W	0x0000_000F	<a href="#">10.3.8.2/10-46</a>
0x6_0090	WHOAMI—Who am I register	R	0x0000_0000	<a href="#">10.3.8.3/10-47</a>
0x6_00A0	IACK—interrupt acknowledge register	R	0x0000_0000	<a href="#">10.3.8.4/10-47</a>
0x6_00B0	EOI—End of interrupt register	W	0x0000_0000	<a href="#">10.3.8.5/10-48</a>

### 10.3.1 Global Registers

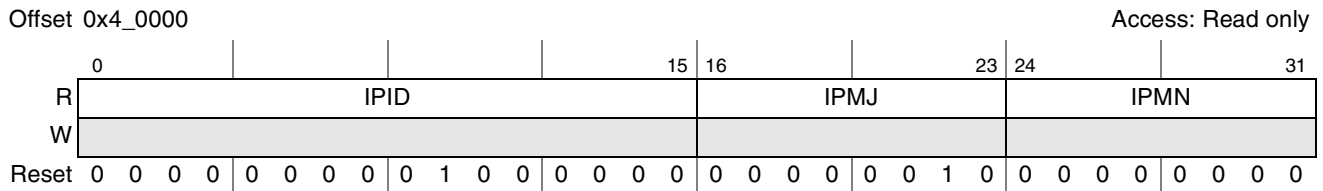
Most registers have one address. Some registers are replicated for each processor in a multiprocessor device. In this case, each processor accesses its separate registers using the same address, the address decoding being sensitive to the processor ID. A copy of the per-CPU registers is available to each processor core at the same physical address, that is, the private access address space. The private access address space acts like an alias to a processor’s own copy of the per-CPU registers. As shown in Figure 10-42, the ID of the processor initiating the read/write transaction is used to determine which processor’s per-CPU registers to access. For more information on per-CPU registers, see [Section 10.3.8, “Per-CPU Registers.”](#)

**NOTE**

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

#### 10.3.1.1 Block Revision Register 1 (BRR1)

The block revision register (BRR1) shown in [Figure 10-3](#) provides information about the IP block.



**Figure 10-3. Block Revision Register 1 (BRR1)**

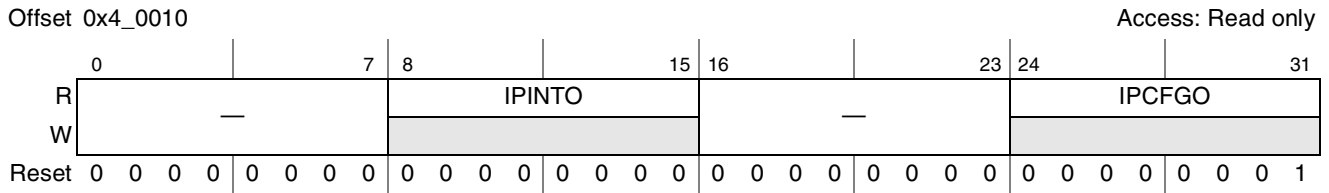
[Table 10-9](#) describes the BRR1 fields.

**Table 10-7. BRR1 Field Descriptions**

Bits	Name	Description
0–15	IPID	IP block ID
16–23	IPMJ	The Major Revision of IP block
24–31	IPMN	The Minor Revision of IP block

### 10.3.1.2 Block Revision Register 2 (BRR2)

The block revision register (BRR2) shown in [Figure 10-4](#) provides information about the IP block integration option and IP block configuration options.



**Figure 10-4. Block Revision Register 2 (BRR2)**

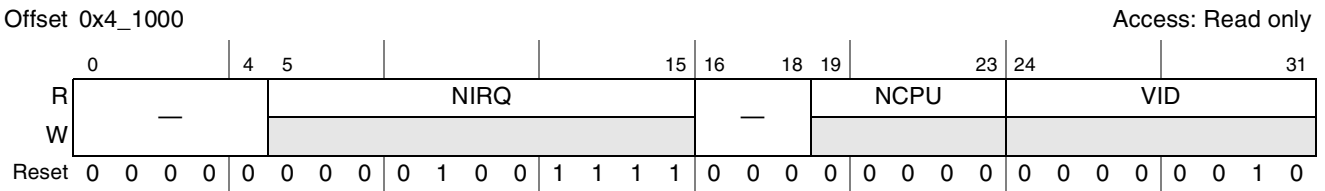
[Table 10-8](#) describes the BRR2 fields.

**Table 10-8. BRR2 Field Descriptions**

Bits	Name	Description
0–7	—	Reserved.
8–15	IPINTO	IP block integration options
16–23	—	Reserved.
24–31	IPCFGO	IP block configuration options

### 10.3.1.3 Feature Reporting Register (FRR)

The feature reporting register (FRR) shown in [Figure 10-5](#) provides information about interrupt and processor configurations. It also informs the programming environment of the controller version.



**Figure 10-5. Feature Reporting Register (FRR)**

[Table 10-9](#) describes the FRR fields.

**Table 10-9. FRR Field Descriptions**

Bits	Name	Description
0–4	—	Reserved.
5–15	NIRQ	Number of interrupts. Contains the binary value of the maximum number of interrupt sources supported minus one. The value is 79 (0x4F) because this device supports 80 interrupts: 12 external sources, 48 internal sources (see <a href="#">Table 10-3</a> ), 4 timer sources, 4 IPI sources, 4 messaging sources, and 8 Shared message signaled sources. A zero in this field corresponds to one source.
16–18	—	Reserved.



### 10.3.1.5 Vendor Identification Register (VIR)

The vendor identification register (VIR) shown in Figure 10-7 has specific read-only information about the vendor and the device revision.

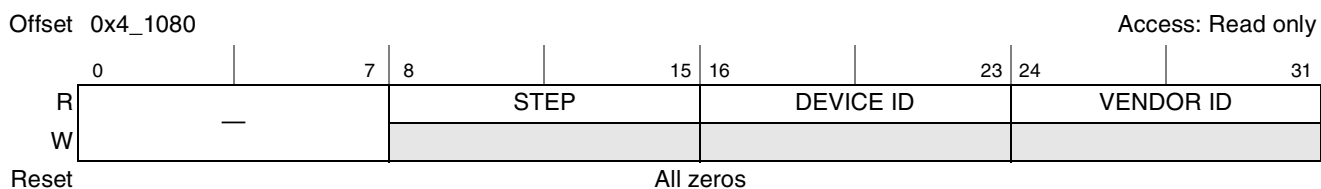


Figure 10-7. Vendor Identification Register (VIR)

Table 10-11 describes the VIR fields.

Table 10-11. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STEP	Stepping. Indicates the silicon revision for this device. Has no meaning when the VENDOR ID value is zero.
16–23	DEVICE ID	Device identification. Vendor-specified identifier for this device. Has no meaning if VENDOR ID = 0.
24–31	VENDOR ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic PIC-compliant device.

### 10.3.1.6 Processor Initialization Register (PIR)

The processor initialization register in the PIC provides a mechanism for software to reset the processor. The *core\_reset* signal to the processor is held active until a zero is written to the processor initialization register field. Thus, PIR should only be written by an external host and the external host should wait at least 100  $\mu$ s to clear this field after writing it. The processor should not attempt to write to PIR because writing to it resets the core; because the core cannot clear the reset field, it would remain in reset indefinitely.

Note that although the architecture was designed to support multiple processing cores, only fields corresponding to processors on the device are implemented, as shown in Figure 10-8. On single-core implementations, only PIR[P0] is implemented.

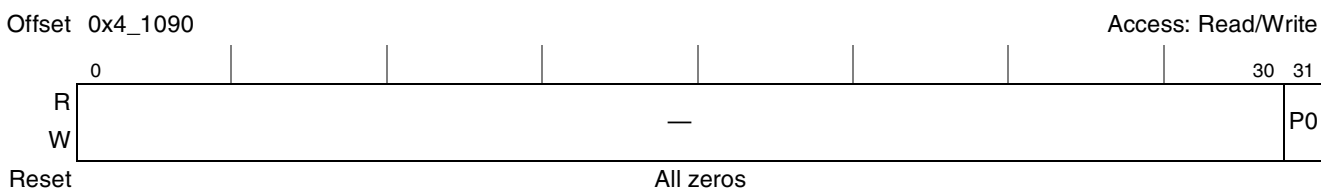


Figure 10-8. Processor Initialization Register (PIR)

Table 10-12 describes the PIR fields.

Table 10-12. PIR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0 core reset. Setting this bit causes the PIC unit to assert the <i>core_reset</i> signal to processor 0 (the e500 core).

### 10.3.1.7 IPI Vector/Priority Registers (IPIVPR<sub>*n*</sub>)

The interprocessor interrupt (IPI) vector/priority registers contain the interrupt vector and priority fields for the four interprocessor interrupt channels as shown in Figure 10-9. There is one IPI vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR<sub>*n*</sub>[A] is set.

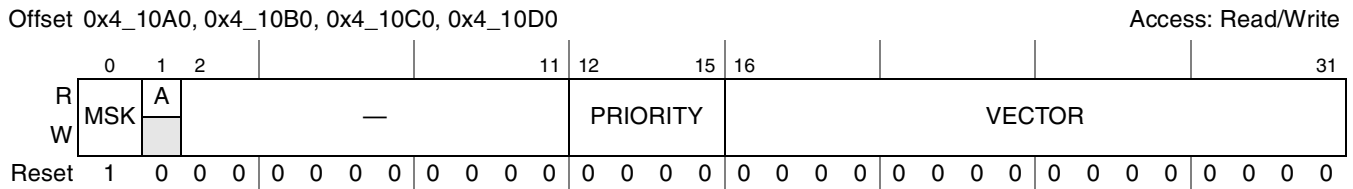


Figure 10-9. IPI Vector/Priority Register (IPIVPR<sub>*n*</sub>)

Table 10-13 describes the IPIVPR<sub>*n*</sub> fields.

Table 10-13. IPIVPR<sub>*n*</sub> Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. Always set following reset. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been reported or is in-service. Note that this field is read only. The values of VECTOR and PRIORITY should not be changed while IPIVPR <sub><i>n</i></sub> [A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt bit for this source in the IPR or ISR is set.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupt reporting from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-48.

### 10.3.1.8 Spurious Vector Register (SVR)

The spurious vector register (SVR), shown in [Figure 10-10](#), contains the 16-bit vector returned to the processor when the IACK register is read for a spurious interrupt.

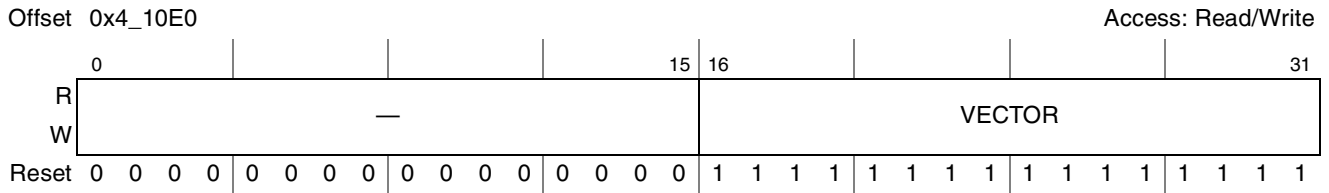


Figure 10-10. Spurious Vector Register (SVR)

Table 10-14 describes the SVR fields.

Table 10-14. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Spurious interrupt vector. Value returned when the interrupt acknowledge (IACK) register is read during a spurious vector fetch. See <a href="#">Section 10.4.3, “Spurious Vector Generation,”</a> for more information about the events and conditions that may cause a spurious vector fetch.

## 10.3.2 Global Timer Registers

This section describes the global timer registers. Note that each timer has four individual configuration registers ( $GTCCR_n$ ,  $GTBCR_n$ ,  $GTVPR_n$ ,  $GTDR_n$ ), but they are only shown once in this section.

### 10.3.2.1 Timer Frequency Reporting Register (TFRR)

The timer frequency reporting register (TFRR), shown in [Figure 10-11](#), is written by software to report the clocking frequency of the PIC timers. Note that although TFRR is read/write, the value of this register is ignored by the PIC unit.

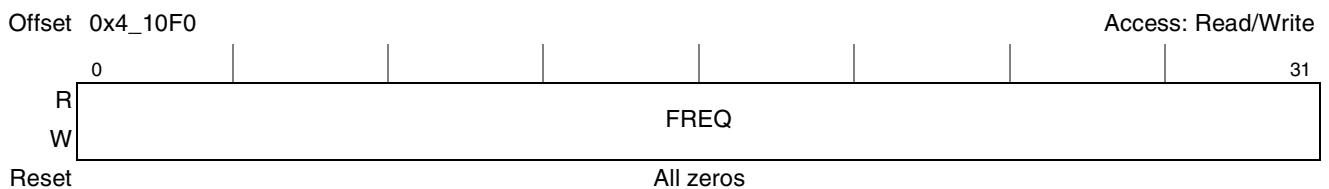


Figure 10-11. Timer Frequency Reporting Register (TFRR)

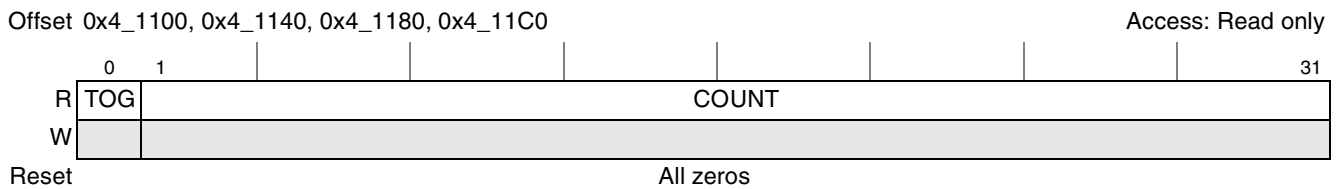
Table 10-15 describes the TFRR register.

**Table 10-15. TFRR Field Descriptions**

Bits	Name	Description
0–31	FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source, identically the core complex bus (CCB) clock, to user software. See <a href="#">Section 4.4.4, "Clocking,"</a> for more details. TFRR is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency set by TCR[CLKR]. See <a href="#">Section 10.3.2.6, "Timer Control Register (TCR)."</a>

### 10.3.2.2 Global Timer Current Count Registers (GTCCR<sub>n</sub>)

The global timer current count registers (GTCCRs), shown in [Figure 10-12](#), contain the current count for each of the four PIC timers.



**Figure 10-12. Global Timer Current Count Registers (GTCCR<sub>n</sub>)**

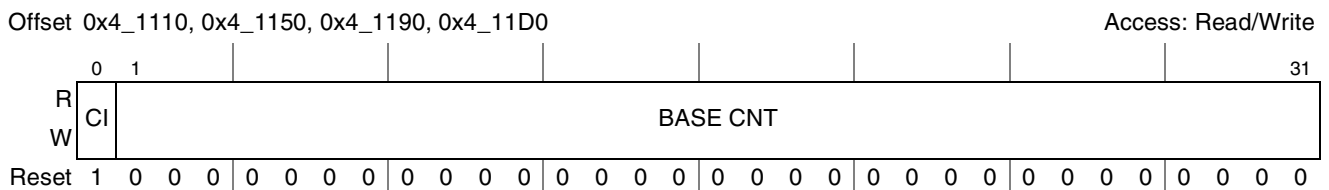
Table 10-16 describes the GTCCR<sub>n</sub> fields.

**Table 10-16. GTCCR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0	TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR <sub>n</sub> [CI] goes from 1 to 0.
1–31	COUNT	Current count. Decrementd while GTBCR <sub>n</sub> [CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding base count register. Cascaded timers are reloaded with either all ones, or the contents of the base count register, depending on the value of TCR[ROVR]. See <a href="#">Section 10.3.2.6, "Timer Control Register (TCR)."</a> for more details.

### 10.3.2.3 Global Timer Base Count Registers (GTBCR<sub>n</sub>)

The global timer base count registers (GTBCRs) contain the base counts for each of the four PIC timers as shown in [Figure 10-13](#). This value is reloaded into the corresponding GTCCR<sub>n</sub> when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR<sub>n</sub>[CI] = 0), the timer generates an interrupt on every timer cycle.



**Figure 10-13. Global Timer Base Count Register (GTBCR<sub>n</sub>)**



Table 10-17 describes the GTBCR $n$  fields.

**Table 10-17. GTBCR $n$  Field Descriptions**

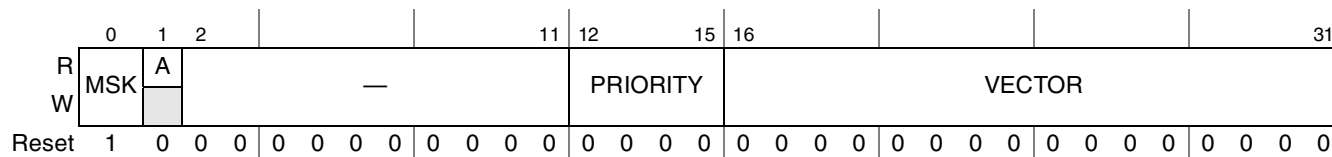
Bits	Name	Description
0	CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31	BASE CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding current count register and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the current count register at the next zero crossing of the current count.

### 10.3.2.4 Global Timer Vector/Priority Registers (GTVPR $n$ )

The global timer vector/priority registers (GTVPRs) contain the interrupt vector and the interrupt priority as shown in Figure 10-14. They also contain the mask and activity fields.

Offset 0x4\_1120, 0x4\_1160, 0x4\_11A0, 0x4\_11E0

Access: Mixed



**Figure 10-14. Global Timer Vector/Priority Register (GTVPR $n$ )**

Table 10-18 describes the GTVPR $n$  fields.

**Table 10-18. GTVPR $n$  Field Descriptions**

Bits	Name	Description
0	MSK	Mask. Inhibits interrupts from this source. 0 Interrupt requests are generated when the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. This field is read-only. 0 No current interrupt activity associated with this source. 1 An interrupt has been requested by the corresponding source or is currently being serviced. The interrupt bit for this source is set in the IPR or ISR. The VECTOR and PRIORITY values should not be changed while the A bit is set.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-48.

### 10.3.2.5 Global Timer Destination Registers (GTDR $n$ )

The global timer destination register controls the destination processor for this timer’s interrupt, as shown in Figure 10-15.

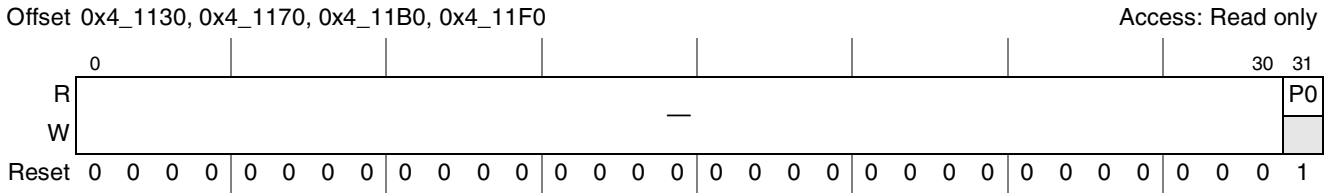


Figure 10-15. Global Timer Destination Registers (GTDR $n$ )

Table 10-19 describes the GTDR $n$  fields.

Table 10-19. GTDR $n$  Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. In a single-core device, internally serviced interrupts are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

### 10.3.2.6 Timer Control Register (TCR)

The timer control register (TCR) shown in Figure 10-17 provides various configuration options such as count frequency and roll-over behavior.

There are two choices for the clock source for the timers: a selectable frequency ratio from the CCB clock, or the RTC signal. The TCR also provides the ability to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer or one 127-bit timer.

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field TCR[ROVR]. Setting a timer’s roll-over field causes its current count register to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF\_FFFF instead of its base count value. Clearing a timer’s associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the settings in TCR[ROVR].

For example, timers 0, 1, and 2 can be cascaded to generate one interrupt every hour. As shown in Table 10-20, given a CCB clock of 333 MHz, letting the timer clock frequency default to 1/8<sup>th</sup> the system clock, (TCR[CLKR] = 0 sets a clock ratio of 8), provides a basic input of 41.625 MHz to timer 0. Setting timer 0 to count 41,625,000 (0x27B\_25A8) timer clock cycles will generate one output per second. Setting both timers 1 and 2 to 59, and cascading all three timers, generates one interrupt every hour from timer 2.

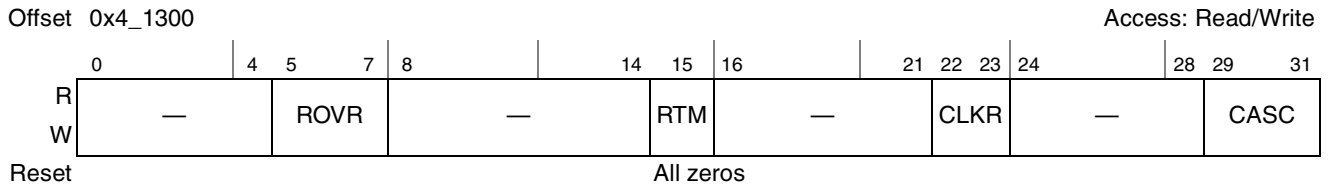
**Table 10-20. Parameters for Hourly Interrupt Timer Cascade Example**

System Clock	Clock Ratio	Timer Clock	Timer 0 Count	Timer 1 Count	Timer 2 Count
333 MHz	1 / 8	41.625 MHz	41.625 x 10 <sup>6</sup> (0x027B_25A8)	59 <sup>1</sup> (0x0000_0036)	59 (0x0000_0036)

<sup>1</sup> Counting down from 59 through 0 requires 60 ticks.

$$(41.625 \times 10^6 \text{ ticks/sec}) \times (60 \text{ sec/min}) \times (60 \text{ min/hr}) = \text{total ticks/hr generating 1 interrupt/hr}$$

**Figure 10-16. Example Calculation for Cascaded Timers**



**Figure 10-17. Timer Control Register (TCR)**

Table 10-21 describes the TCR fields.

**Table 10-21. TCR Field Descriptions**

Bits	Name	Description
0–4	—	Reserved
5–7	ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5–7 correspond to timers 2–0. Note that global timer 3 always reloads with its base count register. 0 Timer does not roll over. When the count reaches zero, current count register is reloaded with the base count register value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, current count register is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF). 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF). 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF). 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF). 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF). 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF). 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14	—	Reserved
15	RTM	Real time mode. Specifies the clock source for the PIC timers. 0 Timer clock frequency is a ratio of the frequency of the platform (CCB) clock as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the PIC timers. If this bit is set, the CLKR field has no meaning.
16–21	—	Reserved

**Table 10-21. TCR Field Descriptions (continued)**

Bits	Name	Description
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the platform (CCB) clock. The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28	—	Reserved
29–31	CASC	Cascade timers. Specifies the output of particular global timers as input to others. 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

### 10.3.3 External, $\overline{\text{IRQ\_OUT}}$ , and Critical Interrupt Summary Registers

The summary registers indicate the interrupt sources directed to  $\overline{\text{IRQ\_OUT}}$  or  $\overline{\text{cint}}$ . Summary register bits are cleared when the corresponding interrupt that caused a bit to be set is negated. Note that only level-sensitive interrupts can be directed to  $\overline{\text{IRQ\_OUT}}$  or  $\overline{\text{cint}}$ .

The  $\overline{\text{IRQ\_OUT}}$  summary registers, shown in [Figure 10-19](#) through [Figure 10-21](#) contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to  $\overline{\text{IRQ\_OUT}}$  (that is, if the corresponding  $\text{xIDR}[\text{EP}]$  is set).

The critical interrupt summary registers shown in [Figure 10-22](#) through [Figure 10-24](#) contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to the processor’s critical interrupt signal  $\overline{\text{cint}}$  (if the CI field in its corresponding destination register is set). The summary register bits are cleared when the corresponding interrupt that caused a bit to be set is negated.

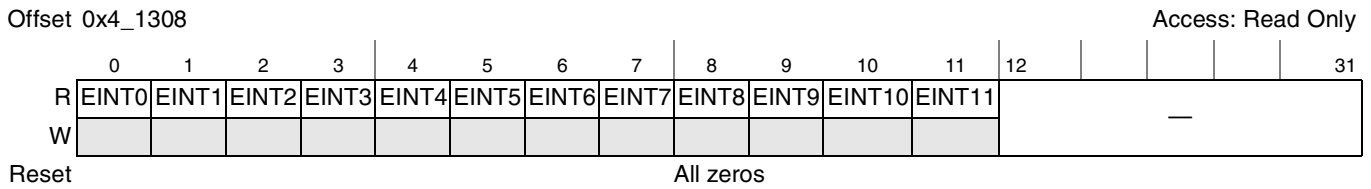
Note that unused  $\text{IRQ}_n$  signals may be used as general-purpose inputs. The external interrupt summary register (ERQSR) can be used to monitor these signals. See [Section 10.3.3.1, “External Interrupt Summary Register \(ERQSR\),”](#) and [Section 19.5.2, “General-Purpose I/O Signals,”](#) for more information.

#### 10.3.3.1 External Interrupt Summary Register (ERQSR)

**NOTE**

The fields in ERQSR report only the current state of  $\text{IRQ}_0$ – $\text{IRQ}_{11}$ . These fields were designed to work with level-sensitive interrupts; the values returned for edge-sensitive interrupts may be unreliable.

Figure 10-18 shows the ERQSR fields.



**Figure 10-18. External Interrupt Summary Register (ERQSR)**

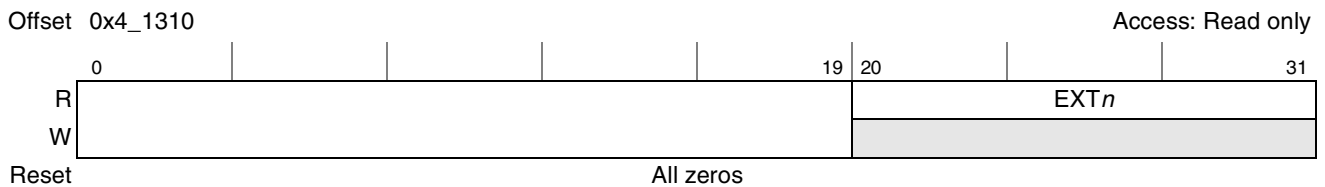
Table 10-22 describes the ERQSR fields.

**Table 10-22. ERQSR Field Descriptions**

Bits	Name	Description
0–11	EINT $n$	External interrupt pin $n$ status. 0 External interrupt pin $n$ is not active. 1 External interrupt pin $n$ is active.
12–31		Reserved

### 10.3.3.2 $\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0)

Figure 10-19 shows the IRQSR0 fields.



**Figure 10-19.  $\overline{\text{IRQ\_OUT}}$  Summary Register 0 (IRQSR0)**

Table 10-23 describes the IRQSR0 fields.

**Table 10-23. IRQSR0 Field Descriptions**

Bits	Name	Description												
0–19	—	Reserved												
20–31	EXT $n$	External interrupts 0–11. Each bit corresponds to a different interrupt according to the following: <table style="margin-left: 20px; border: none;"> <tr><td>Bit</td><td>Interrupt</td></tr> <tr><td>20</td><td>IRQ0</td></tr> <tr><td>21</td><td>IRQ1</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>31</td><td>IRQ11</td></tr> </table> 0 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ . 1 The corresponding interrupt is active and directed to $\overline{\text{IRQ\_OUT}}$ (if the corresponding xIDR[EP] is set).	Bit	Interrupt	20	IRQ0	21	IRQ1	.	.	.	.	31	IRQ11
Bit	Interrupt													
20	IRQ0													
21	IRQ1													
.	.													
.	.													
31	IRQ11													

### 10.3.3.3 $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Figure 10-20 shows the IRQSR1 fields.

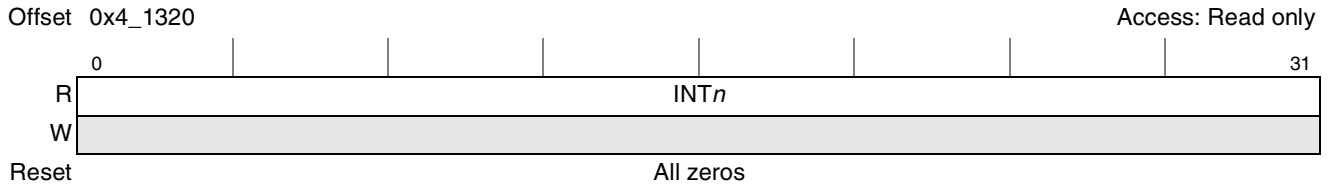


Figure 10-20.  $\overline{\text{IRQ\_OUT}}$  Summary Register 1 (IRQSR1)

Table 10-24 describes the IRQSR1 fields.

Table 10-24. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT <sub>n</sub>	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ . 1 The corresponding interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding xIDR[EP] is set).

### 10.3.3.4 $\overline{\text{IRQ\_OUT}}$ Summary Register 2 (IRQSR2)

Figure 10-21 shows the IRQSR2 fields.

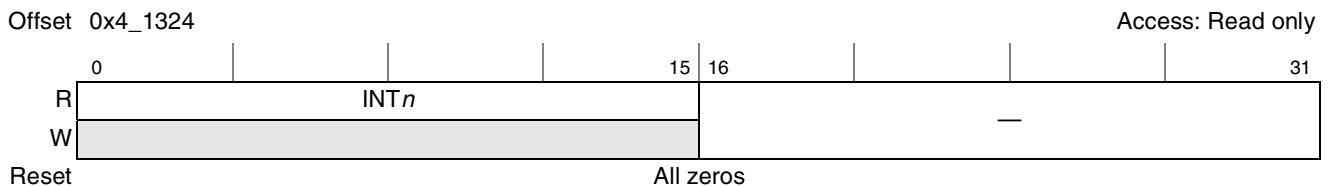


Figure 10-21.  $\overline{\text{IRQ\_OUT}}$  Summary Register 2 (IRQSR2)

Table 10-25 describes the IRQSR2 fields.

Table 10-25. IRQSR2 Field Descriptions

Bits	Name	Description
0–15	INT <sub>n</sub>	Internal interrupts 32–47 status. Bit 0 represents INT32. Bit 15 represents INT47. 32 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ . 47 The corresponding interrupt is active and directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding xIDR[EP] is set).
16–31	—	Reserved

### 10.3.3.5 Critical Interrupt Summary Register 0 (CISR0)

Figure 10-22 shows the CISR0 fields.

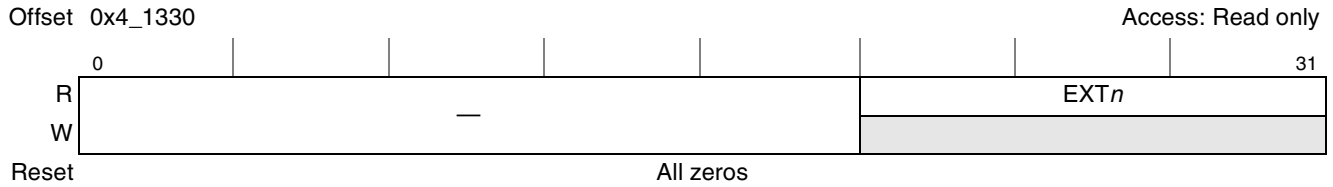


Figure 10-22. Critical Interrupt Summary Register 0 (CISR0)

Table 10-26 describes the CISR0 fields.

Table 10-26. CISR0 Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	EXT $n$	External interrupts 0–11. Bit 20 represents IRQ0. Bit 31 represents IRQ11. 0 The corresponding interrupt is not active or not directed to $\overline{cint}$ . 1 The corresponding interrupt is active and is directed to the $\overline{cint}$ (if the corresponding xIDR[C] is set).

### 10.3.3.6 Critical Interrupt Summary Register 1 (CISR1)

Figure 10-23 shows the CISR1 fields.

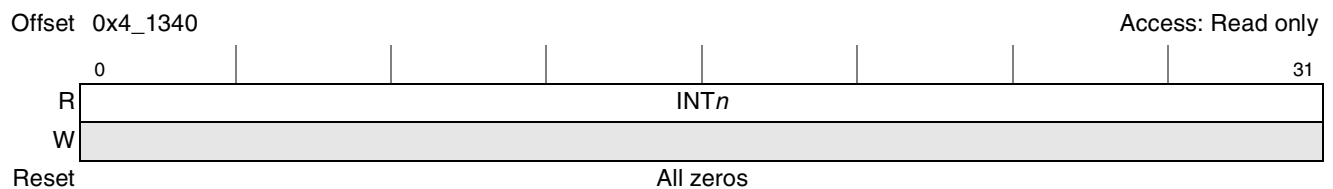


Figure 10-23. Critical Interrupt Summary Register 1 (CISR1)

Table 10-27 describes CISR1 register.

Table 10-27. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT $n$	Internal interrupts 0–31. Bit 0 represents INT0. Bit 31 represents INT31. 0 Corresponding interrupt is not active or not directed to $\overline{cint}$ . 1 The corresponding interrupt is active and is directed to the $\overline{cint}$ (if the corresponding xIDR[C] is set).

### 10.3.3.7 Critical Interrupt Summary Register 2 (CISR2)

Figure 10-24 describes the CISR2 register.

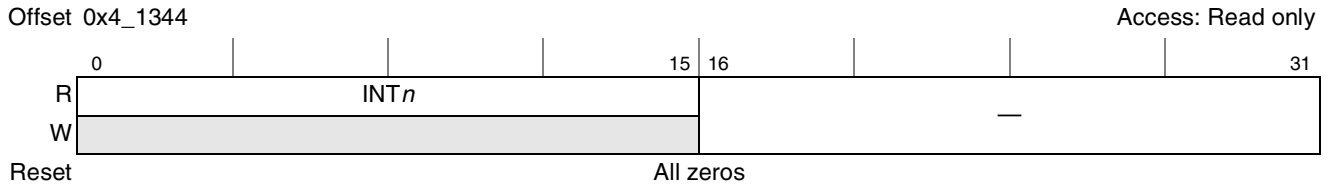


Figure 10-24. Critical Interrupt Summary Register 2 (CISR2)

Table 10-28 shows CISR2 fields.

Table 10-28. CISR2 Field Descriptions

Bits	Name	Description
0–15	INT $n$	Internal interrupts 32–47. Bit 0 represents INT32. Bit 15 represents INT47. 0 Corresponding interrupt is not active or not directed to $\overline{cint}$ . 1 The corresponding interrupt is active and is directed to the $\overline{cint}$ (if the corresponding xIDR[CI] is set).
16–31	—	Reserved

### 10.3.4 Performance Monitor Mask Registers (PMMRs)

The twelve performance monitor mask registers consist of four sets of three 32-bit registers—PM $n$ MR0, PM $n$ MR1, and PM $n$ MR2. Each set can be configured to select one interrupt source (IPI, timer, message, or external) to generate a performance monitor event. The performance monitor can be configured to track this event in the performance monitor local control registers. See Section 20.3.2.2, “Performance Monitor Local Control Registers (PMLCAn, PMLCBn).”

#### 10.3.4.1 Performance Monitor $n$ Mask Registers 0 (PM $n$ MR0)

Figure 10-25 shows the PM $n$ MR0 registers. Each PM $n$ MR0 register is associated with a PM $n$ MR1 and a PM $n$ MR2 register. Because each unreserved bit in the 96-bit set (PM $n$ MR0/1/2) specifies a different interrupt, only one bit in each set can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

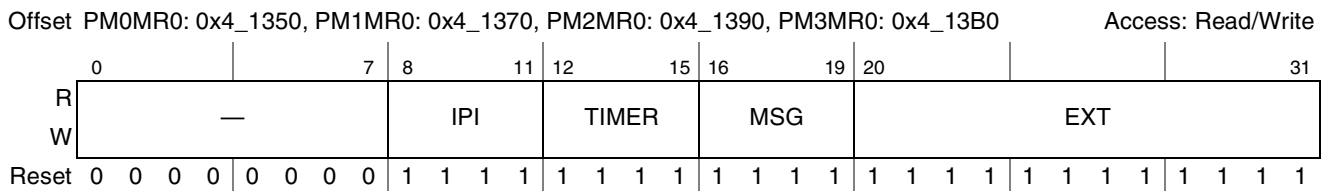


Figure 10-25. Performance Monitor  $n$  Mask Registers 0 (PM $n$ MR0)



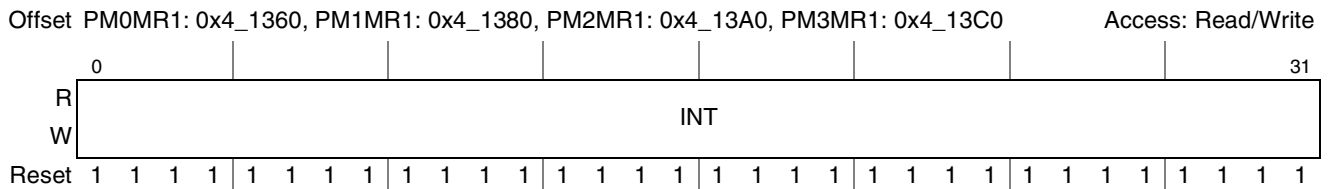
Table 10-29 describes the PM $n$ MR0 fields.

**Table 10-29. PM $n$ MR0 Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–11	IPI	IPI interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15	TIMER	Timer interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19	MSG	Message interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31	EXT	External interrupts IRQ[0:11] 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

#### 10.3.4.2 Performance Monitor $n$ Mask Registers 1 (PM $n$ MR1)

Figure 10-26 shows the PM0MR1, PM1MR1, PM2MR1, and PM3MR1 registers.



**Figure 10-26. Performance Monitor  $n$  Mask Registers 1 (PM $n$ MR1)**

Table 10-30 describes the PM $n$ MR1 fields.

**Table 10-30. PM $n$ MR1 Field Descriptions**

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

### 10.3.4.3 Performance Monitor *n* Mask Registers 2 (PM<sub>*n*</sub>MR2)

Figure 10-27 shows the PM0MR2, PM1MR2, PM2MR2, and PM3MR2 registers.

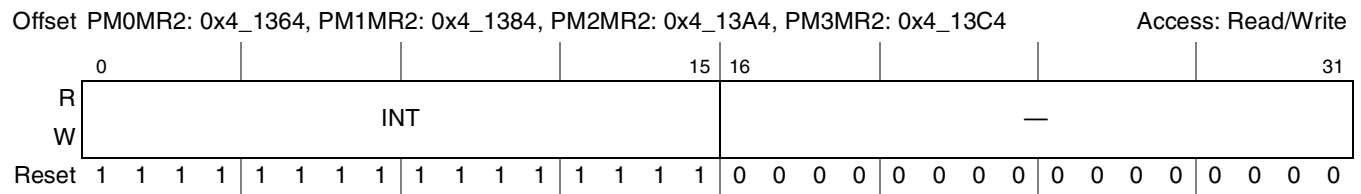


Figure 10-27. Performance Monitor *n* Mask Registers 2 (PM<sub>*n*</sub>MR2)

Table 10-30 describes the PM<sub>*n*</sub>MR2 fields.

Table 10-31. PM<sub>*n*</sub>MR2 Field Descriptions

Bits	Name	Description
0–15	INT	Internal interrupts 32–48 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–31	—	Reserved, should be cleared.

## 10.3.5 Message Registers

Writing to one of the four message registers (MSGR0–MSGR3) causes a messaging interrupt to the processor. Reading this register clears the messaging interrupt. Note that a messaging interrupt can also be cleared by writing a one to the corresponding status field of the PIC message status register (MSR), shown in Figure 10-30.

### 10.3.5.1 Message Registers (MSGR0–MSGR3)

The message registers (MSGR0–MSGR3), shown in Figure 10-28, contain a 32-bit message.

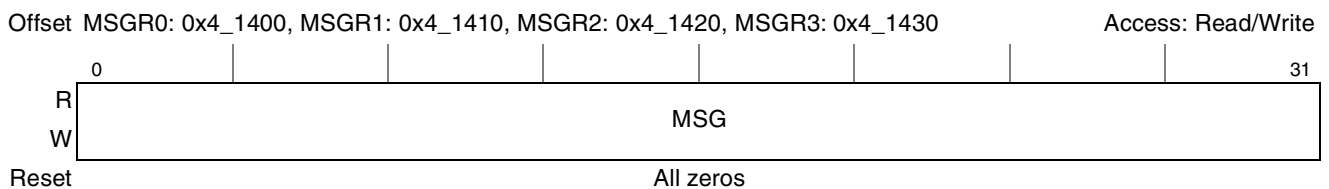


Figure 10-28. Message Registers (MSGRs)

Table 10-32 describes the MSGR registers.

Table 10-32. MSGR<sub>*n*</sub> Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data.

### 10.3.5.2 Message Enable Register (MER)

The MER shown in Figure 10-29 contains the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

When bits in MER are set to mask message interrupts, an interrupt is not generated if the message register is written while it is masked in MER and the MER bit is then cleared. To mask the interrupt without loss, set MIVPR<sub>n</sub>[MSK]. (See Section 10.3.7.5, “Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3).”) MER should be set to 0x0000\_000F at reset and be left unchanged during normal operation.

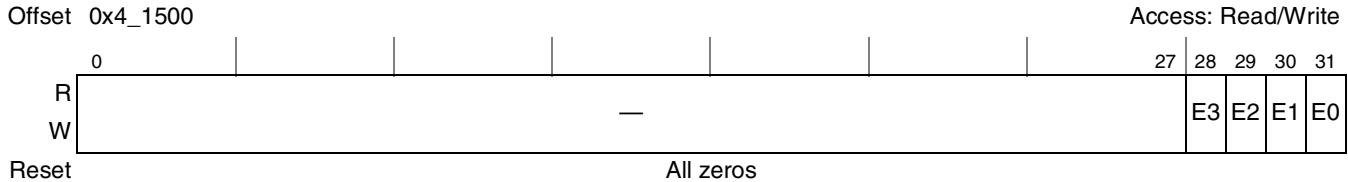


Figure 10-29. Message Enable Register (MER)

Table 10-33 describes the MER fields.

Table 10-33. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	E3–E0	Enable <i>n</i> . Used to enable interrupt generation for MSGR <sub>n</sub> . 0 Interrupt generation for MSGR <sub>n</sub> disabled. 1 Interrupt generation for MSGR <sub>n</sub> enabled.

### 10.3.5.3 Message Status Register (MSR)

The PIC message status register (MSR) shown in Figure 10-30 contains status bits for each message register. The status bit is set when the corresponding messaging interrupt is active. Writing a one to a status bit clears the corresponding message interrupt and the status bit.

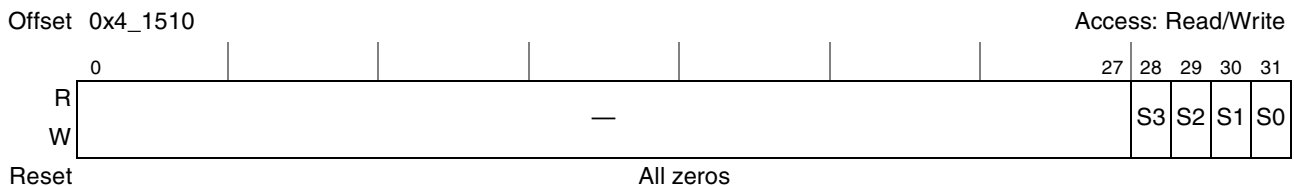


Figure 10-30. Message Status Register (MSR)

Table 10-34 describes the MSR fields.

**Table 10-34. MSR Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	S3–S0	Status <i>n</i> . Reports status of messaging interrupt 3. Writing a 1 clears this field. 0 Messaging interrupt <i>n</i> is not active. 1 Messaging interrupt <i>n</i> is active.

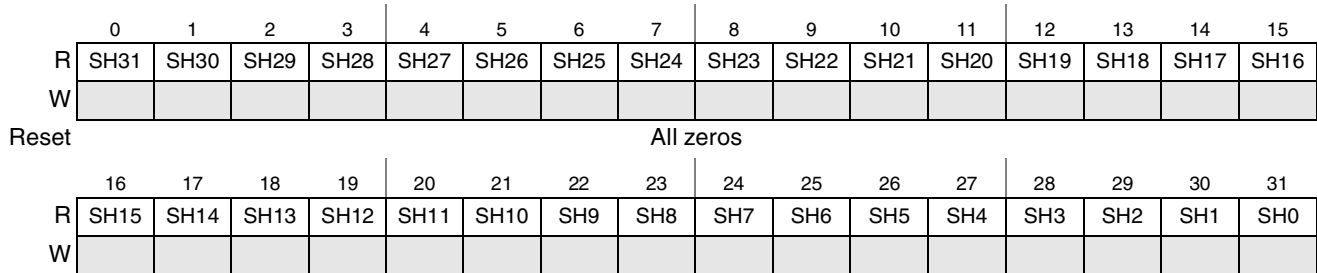
### 10.3.6 Shared Message Signaled Registers

This section contains the description of all of the shared message signaled interrupt registers.

#### 10.3.6.1 Shared Message Signaled Interrupt Registers (MSIRs)

The shared message signaled interrupt registers shown in Figure 10-31 indicate which of the interrupt sources sharing the message register have pending interrupts. Up to 32 sources can share any individual message register. These registers are cleared when read. A write to these registers has no effect.

Offset 0x4\_1600, 0x4\_1610, 0x4\_1620, 0x4\_1630, 0x4\_1640, 0x4\_1650, 0x4\_1660, 0x4\_1670 Access: Special



Reset All zeros (Note: After reset, read the MSIRs to ensure that any interrupts previously pending are cleared).

**Figure 10-31. Shared Message Signaled Interrupt Register (MSIRs)**

Table 10-35 describes the bits of the MSIRs.

**Table 10-35. MSIRs Field Descriptions**

Bits	Name	Description
31–0	SH <i>n</i>	Message sharer <i>n</i> ( <i>n</i> = 31–0) has a pending interrupt.

#### 10.3.6.2 Shared Message Signaled Interrupt Status Register (MSISR)

This register shown in Figure 10-32 contains the status bits for the shared message signaled interrupts. The status bit is set to 1 when the corresponding shared message signaled interrupt is active. The status bit is 0 if all the corresponding shared interrupt sources are cleared in the shared message signaled interrupt register (MSIR). This register is read-only.

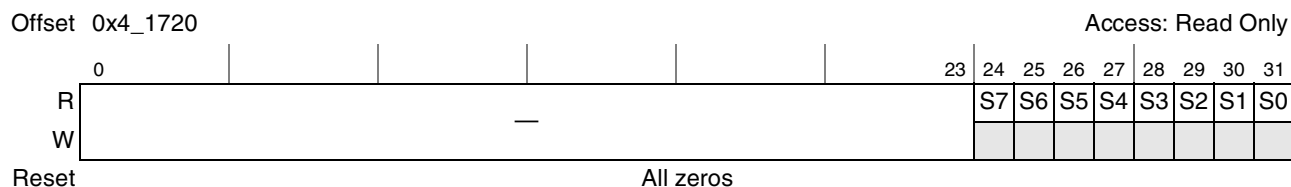


Figure 10-32. Shared Message Signaled Interrupt Status Register (MSISR)

Table 10-36 describes the bits of the MSISR.

Table 10-36. MSISR Field Descriptions

Bits	Name	Description
0–23	—	Reserved.
24–31	S7–S0	Status <i>n</i> . Set when shared message signaled interrupt <i>n</i> is active

### 10.3.6.3 Shared Message Signaled Interrupt Index Register (MSIIR)

MSIIR provides a mechanism for setting an interrupt in the shared message signaled interrupt registers. There are two fields. When MSIIR is written, MSIIR[SRS] selects the register in which an interrupt bit is to be set; MSIIR[IBS] selects the shared interrupt field in the selected MSIR register to be set. This register is primarily intended to support PCI Express MSIs. This register is write-only.



Figure 10-33. Shared Message Signaled Interrupt Index Register (MSIIR)

Table 10-37 describes the bits of the MSIIRs.

Table 10-37. MSIIR Field Descriptions

Bits	Name	Description
0–2	SRS	Shared interrupt register select—Select the shared message signaled interrupt register 000 Shared message signaled interrupt register 0 001 Shared message signaled interrupt register 1 ... 111 Shared message signaled interrupt register 7
3–7	IBS	Interrupt bit select—Selects the bit to set in the MSIR 00000 Set field SH0 (bit 31) 00001 Set filed SH1 (bit 30) ... 11111 Set field SH31 (bit 0)
8–31	—	Reserved

### 10.3.6.4 Shared Message Signaled Interrupt Vector/Priority Register (MSIVPR<sub>n</sub>)

The shared message signaled interrupt vector/priority registers have the same field and format as the GTVPRs. These registers are read/write.

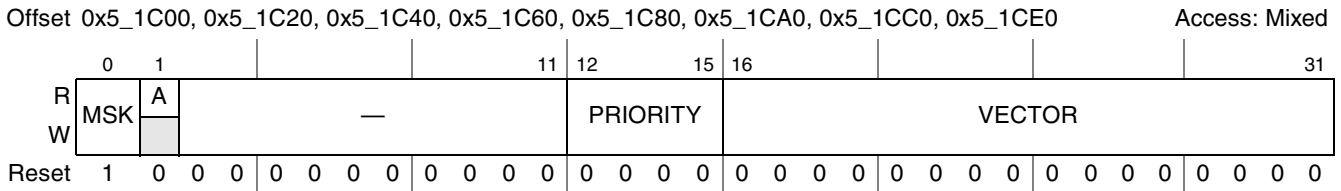


Figure 10-34. Shared Message Signaled Interrupt Vector/Priority Register (MSIVPR<sub>n</sub>)

Table 10-38 describes the bits of the MSIVPR<sub>n</sub>.

Table 10-38. MSIVPR<sub>n</sub> Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 If the mask bit is cleared while the corresponding IPR bit is set, an interrupt request is generated. 1 Further interrupts from this source are disabled. This bit is always set to a 1 following reset.
1	A	Activity (read only). Indicates whether an interrupt has been requested or that it is in-service. 0 No current interrupt activity associated with this source. 1 The interrupt bit for this source in the IPR or ISR is set. The PRIORITY value should not be changed while the A bit is set.
2–11	—	Reserved.
12–15	PRIORITY	Priority. Sets interrupt priority. The highest priority is 15; the lowest is 0 (interrupt generation disabled).
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is examined and the interrupt associated with this vector has been requested.

### 10.3.6.5 Shared Message Signaled Interrupt Destination Register (MSIDR<sub>n</sub>)

The shared message signaled interrupt destination registers contain the destination bits for the shared message signaled interrupt. A shared message signaled interrupt can be directed to one of the processors by setting the appropriate bit in the shared message signaled interrupt destination register. Only one of the bits corresponding to destination processors may be set. The behavior is more than one bit is set is not defined. This register also contains the external pin and critical interrupt fields that can be programmed to direct the interrupt to the external pin or critical interrupt pin of the processor, respectively.

These registers are read/write, as shown in Figure 10-35.

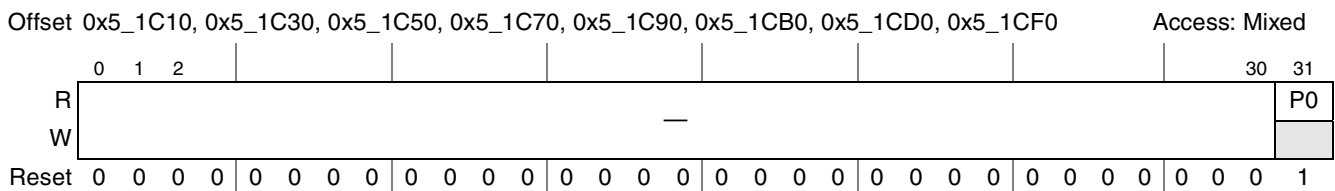


Figure 10-35. Shared Message Signaled Interrupt Destination Registers (MSIDR<sub>n</sub>)

Table 10-39 describes the bits of the MSIDR $n$ .

**Table 10-39. MSIDR $n$  Field Descriptions**

Bits	Name	Description
0–30	—	Reserved.
31	P0	Processor 0. Set to 1 directs the interrupt to processor 0.

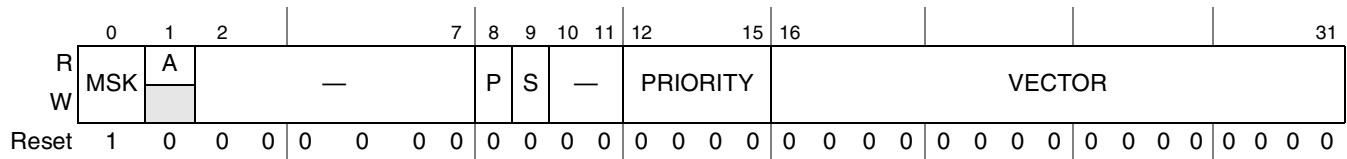
### 10.3.7 Interrupt Source Configuration Registers

The interrupt source configuration registers control the source of each interrupt, specifying parameters such as the interrupting event, signal polarity, and relative priority.

#### 10.3.7.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The external interrupt vector/priority registers (EIVPRs) contain polarity and sense fields for the external interrupts caused by the assertion of any of IRQ[0:11]. The format of the EIVPRs is shown in Figure 10-36.

Offset EIVPR0: 0x5\_0000, EIVPR1: 0x5\_0020, EIVPR2: 0x5\_0040, EIVPR3: 0x5\_0060, EIVPR4: 0x5\_0080, Access: Mixed  
 EIVPR5: 0x5\_00A0, EIVPR6: 0x5\_00C0, EIVPR7: 0x5\_00E0, EIVPR8: 0x5\_0100, EIVPR9: 0x5\_0120,  
 EIVPR10: 0x5\_0140, EIVPR11: 0x5\_0160



**Figure 10-36. External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)**

Table 10-40 describes the EIVPR fields.

**Table 10-40. EIVPR $n$  Field Descriptions**

Bits	Name	Description
0	MSK	Mask. Masks interrupts from this source. 0 If the corresponding IPR bit is set an interrupt request is generated. 1 Interrupts from this source are disabled.
1	A	Activity. A read-only field indicating that an interrupt has been requested or is in-service. Note: The P (polarity), S (sense), VECTOR and PRIORITY values should not be changed while the corresponding interrupt is active, that is, while EIVPR $n$ [A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt bit for this source is set in the IPR or ISR.
2–7	—	Reserved.
8	P	Polarity. Specifies the polarity for the external interrupt. 0 Polarity is active-low or negative edge triggered. 1 Polarity is active-high or positive edge-triggered.
9	S	Sense. Specifies the sense for external interrupts. 0 The external interrupt is edge sensitive. 1 The external interrupt is level-sensitive.

**Table 10-40. EIVPR<sub>n</sub> Field Descriptions (continued)**

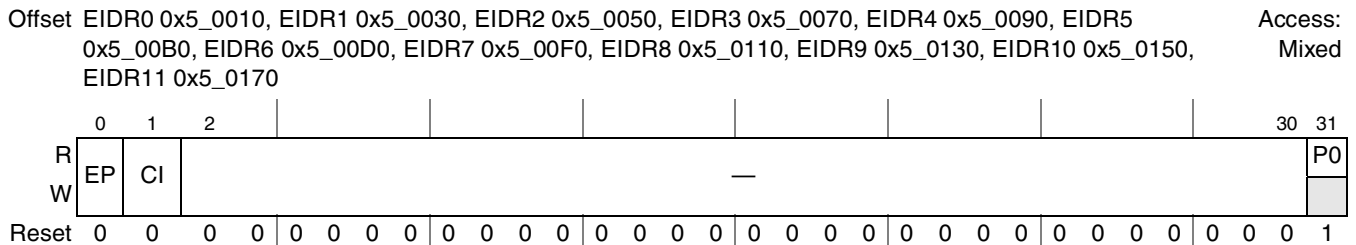
Bits	Name	Description
10–11	—	Reserved.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. Contains the value returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in <a href="#">Figure 10-48</a> .

### 10.3.7.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The external interrupt destination registers (EIDRs), shown in [Figure 10-37](#), control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. All external interrupts are directed to the processor 0 interrupt, *int*, by default. External interrupts can be selectively directed to *IRQ\_OUT* or to the processor 0 critical interrupt signal, *cin*, instead of *int* by writing to the appropriate EIDR<sub>n</sub> fields.

#### NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.



**Figure 10-37. External Interrupt Destination Registers (EIDRs)**

[Table 10-41](#) describes the EIDR fields. Because external interrupts can be channeled only to processor 0, the P0 bit is permanently set. As shown in [Figure 10-48](#), if either the CI or EP bits are set, the interrupt is not sent to the processor’s interrupt input.

The EP or CI fields must be set only for level-sensitive interrupts. Setting these fields for edge-sensitive interrupts does not provide reliable interrupt response.

**Table 10-41. EIDR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0	EP	External pin. Allows external interrupt to be serviced externally. 0 External interrupt is serviced internally with <i>int</i> signal to the processor core. 1 External interrupt is directed to <i>IRQ_OUT</i> for external service.
1	CI	Critical interrupt. 0 External interrupt is serviced internally with <i>int</i> signal to the processor core. 1 External interrupt is directed to processor 0 as a critical interrupt with the <i>cin</i> signal.



**Table 10-41. EIDR<sub>n</sub> Field Descriptions (continued)**

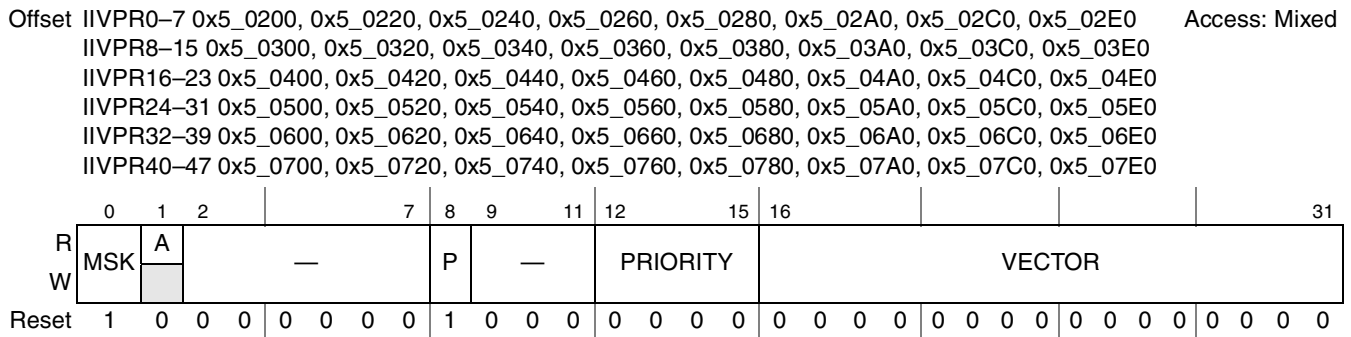
Bits	Name	Description
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. P0 is meaningful only in a multi-core device. In a single-core device, all interrupts that are serviced internally are directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

### 10.3.7.3 Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR47)

The internal interrupt vector/priority registers (IIVPRs), shown in [Figure 10-38](#), have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in [Table 10-42](#). These interrupts are all level-sensitive.

#### NOTE

Because all internal interrupts are active-high, clearing any IIVPR<sub>n</sub> polarity field disables that interrupt. Care should be taken to ensure this field is not inadvertently corrupted when loading or reloading IIVPRs with priority, mask, or vector data.



**Figure 10-38. Internal Interrupt Vector/Priority Registers (IIVPRs)**

[Table 10-42](#) describes the IIVPR fields.

**Table 10-42. IIVPR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while IIVPR <sub>n</sub> [A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved

**Table 10-42. IIVPRn Field Descriptions (continued)**

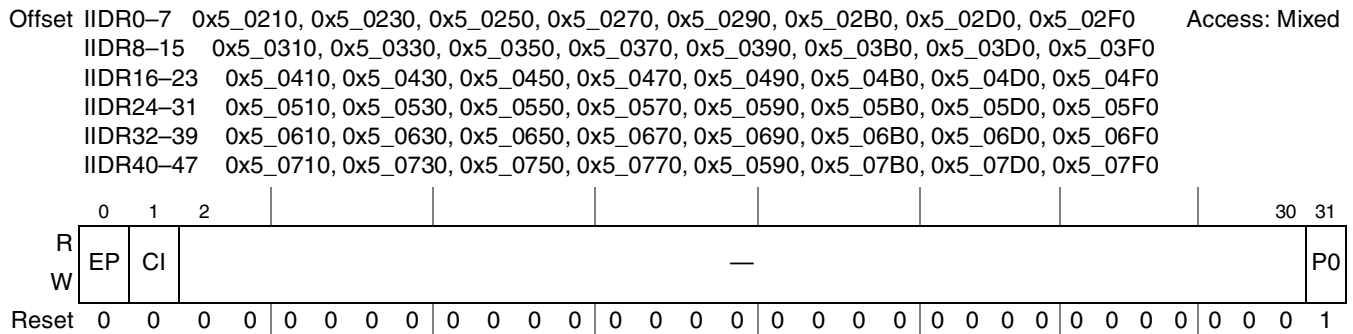
Bits	Name	Description
8	P	Polarity. Specifies the polarity for the internal interrupt. Note: Because all internal interrupts are active-high, clearing this bit disables the interrupt. 0 Interrupt polarity is active-low. This value disables the interrupt. 1 Interrupt polarity is active-high. This is the reset value and should not be changed.
9–11	—	Reserved.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-48.

### 10.3.7.4 Internal Interrupt Destination Registers (IIDR0–IIDR47)

Internal interrupt destination registers (IIDRs), shown in Figure 10-39, have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources listed in Table 10-43. Like external interrupts, internal interrupts can be directed only to processor 0.

#### NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.



**Figure 10-39. Internal Interrupt Destination Registers (IIDRs)**

Table 10-43 describes the IIDR fields.

**Table 10-43. IIDRn Field Descriptions**

Bits	Name	Description
0	EP	External pin. Allows internal interrupt to be serviced externally. 0 Internal interrupt is serviced internally with $\overline{int}$ signal to the processor core. 1 Internal interrupt is directed to $\overline{IRQ\_OUT}$ for external service. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
1	CI	Critical interrupt. 0 Internal interrupt is serviced internally with $\overline{int}$ signal to the processor core. 1 Internal interrupt is directed to the processor 0 as a critical interrupt with the $\overline{cint}$ signal. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.

Table 10-43. IIDR<sub>n</sub> Field Descriptions (continued)

Bits	Name	Description
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. For a single-core device, interrupts serviced internally are directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

### 10.3.7.5 Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3)

The messaging interrupt vector/priority registers (MIVPRs), shown in [Figure 10-40](#), have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

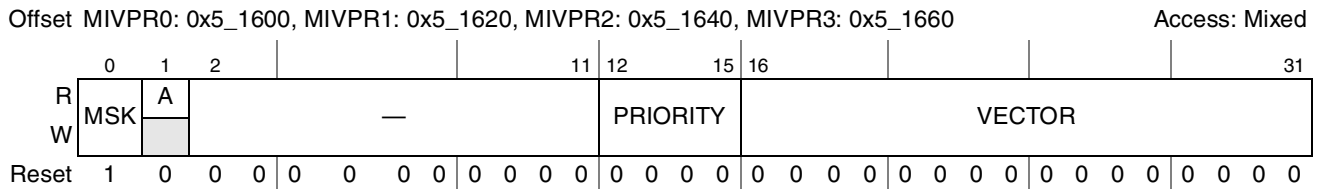


Figure 10-40. Messaging Interrupt Vector/Priority Registers (MIVPRs)

[Table 10-44](#) describes the MIVPR fields.

 Table 10-44. MIVPR<sub>n</sub> Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while MIVPR <sub>n</sub> [A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in <a href="#">Figure 10-48</a> .

### 10.3.7.6 Messaging Interrupt Destination Registers (MIDR0–MIDR3)

The messaging interrupt destination registers (MIDRs), shown in [Figure 10-41](#), control the destination for the messaging interrupts. MIDR enables the user to direct the interrupt to either the external interrupt output pin ( $\overline{\text{IRQ\_OUT}}$ ), the core's critical interrupt input ( $\overline{\text{cint}}$ ), or to its normal interrupt input ( $\overline{\text{int}}$ ).

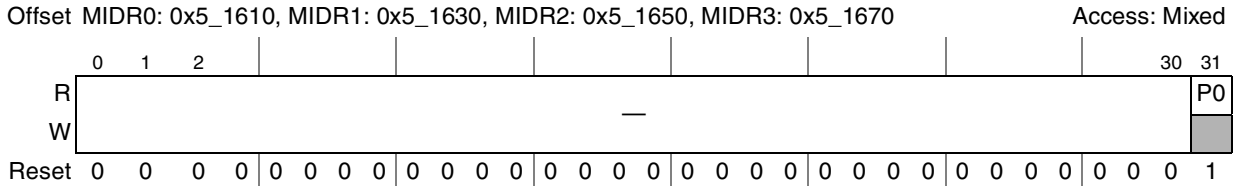


Figure 10-41. Messaging Interrupt Destination Registers (MIDRs)

Table 10-45 describes the MIDR fields.

Table 10-45. MIDR<sub>n</sub> Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because this is a single-core device, all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

### 10.3.8 Per-CPU Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To fully comply with the OpenPIC specification, the PIC incorporates several of these multiprocessor capabilities. Because the value of features such as private address space for per-CPU registers and interprocessor interrupts is fully realized only in a multi-core environment, their utility in this single-core device is not intuitive.

The registers in Table 10-46 are called per-CPU registers, because they would be duplicated for each core in a multi-core device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor that initiates the transaction to determine the set of per-CPU registers to access.

Table 10-46. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
IPI 0 dispatch register (IPIDR0)	0x4_0040
IPI 1 dispatch register (IPIDR1)	0x4_0050
IPI 2 dispatch register (IPIDR2)	0x4_0060
IPI 3 dispatch register (IPIDR3)	0x4_0070
Current task priority register (CTPR)	0x4_0080
Who am I register (WHOAMI0)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

These addresses, shown in Table 10-46, appear in the memory map at the same offset for every processor, and are called the private access space. Because this device has only one core, there is only one set of per-CPU registers, each register having two addresses. For example, the CTPR is located normally at 0x6\_0080, and also at the private access address of 0x4\_0080. While this double mapping seems superfluous on a single-core device, the purpose of this feature is to enable user code to execute correctly in an multiprocessor environment without needing to know which CPU it is running on. It is included on this device to simplify the porting of such code.

An example of how the different registers are addressed in a four-core device is illustrated in Figure 10-42. Note that when accessing a register normally, each core sources a different address. However, when accessing the same register using the per-CPU address space, each core sources the same address.

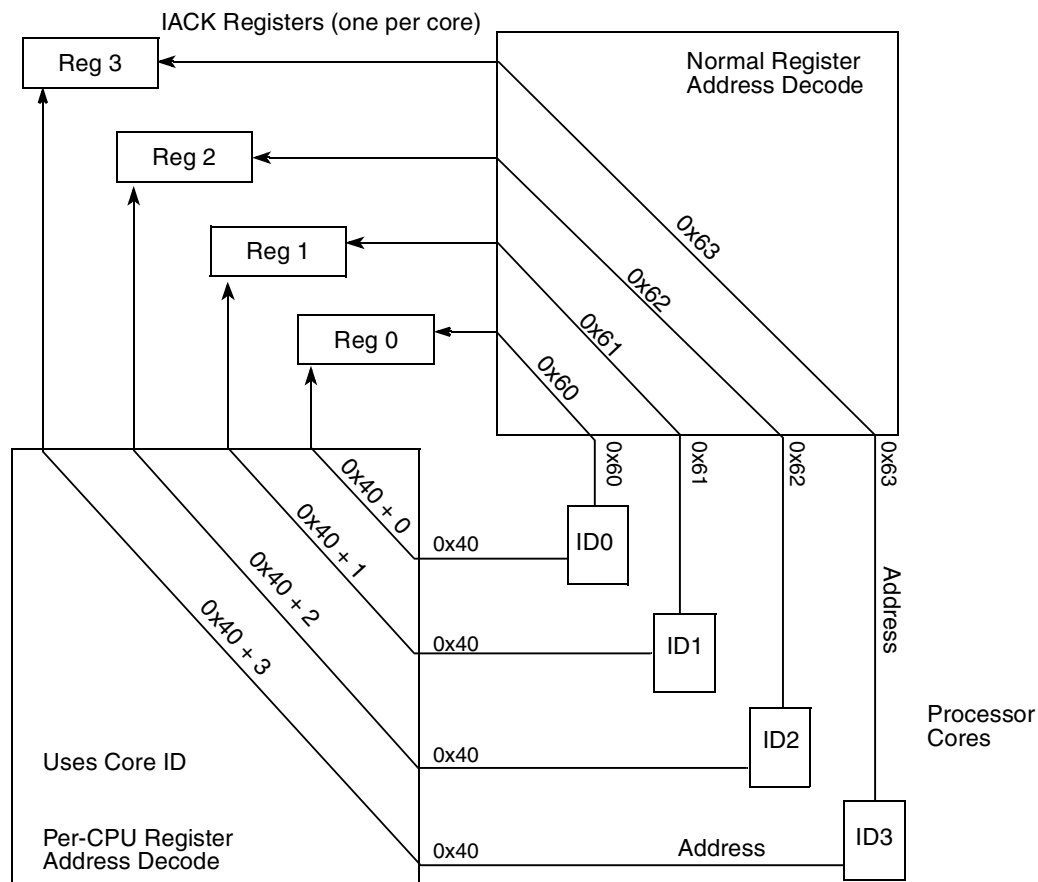
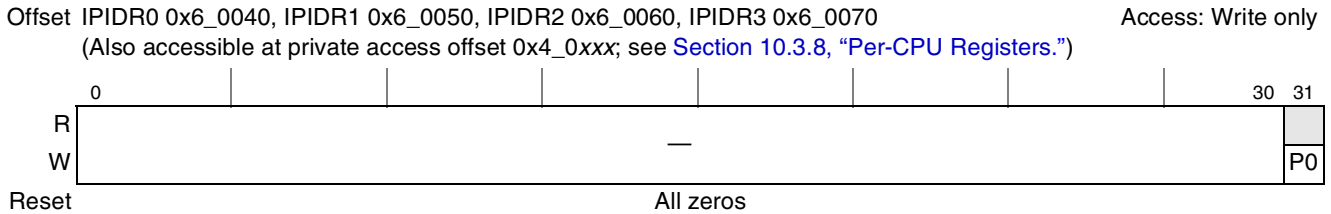


Figure 10-42. Per-CPU Register Address Decoding in a Four-Core Device

### 10.3.8.1 Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

There are four interprocessor interrupt dispatch registers (IPIDRs), one for each IPI channel, as shown in Figure 10-43. Writing to an IPIDR with a bit set causes a self interrupt. While the concept of interprocessor interrupts apparently makes little sense in a single-core device, this feature can serve as a doorbell type interrupt because external bus masters can write to these registers.



**Figure 10-43. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)**

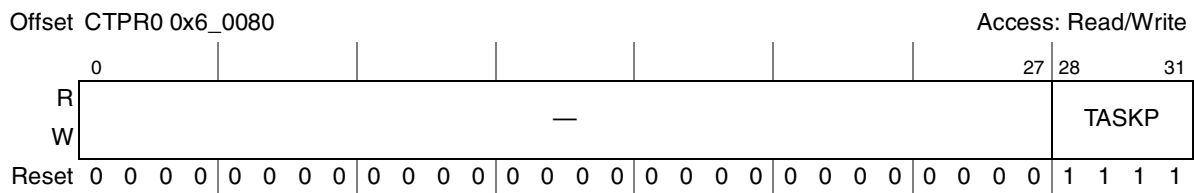
Table 10-42 describes the IPIDR<sub>n</sub> fields.

**Table 10-47. IPIDR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Determines if processor 0 receives the interrupt. (write only). 0 Processor 0 does not receive the interrupt. 1 Directs the interrupt to processor 0.

### 10.3.8.2 Processor Current Task Priority Register (CTPR)

There is one CTPR on this device, shown in Figure 10-44.



**Figure 10-44. Processor Current Task Priority Register (CTPR)**

Software must write the priority of the current processor task in the CTPR. The PIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to the core if the following apply:

- The interrupt is not masked
- The priority of the interrupt is higher than the values in the CTPR and ISR

Priority levels from 0 (lowest) to 15 (highest) are supported. Setting the task priority to 15 masks all interrupts to the processor. Hardware sets the task register to 0x0000\_000F during reset or when the Px field of the processor initialization register is set.

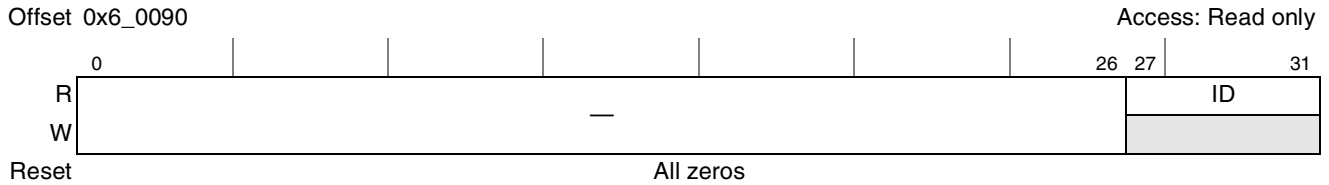
Table 10-48 describes the fields of the CTPR.

**Table 10-48. CTPR Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	TASKP	Task priority. This field is set from 0 to 15, where 15 corresponds to the highest priority for processor tasks. If CTPR[TASKP] = 0xF, no interrupts are signaled to the processor.

### 10.3.8.3 Who Am I Register (WHOAMI)

The processor who am I register (WHOAMI), shown in [Figure 10-45](#), can be read by the processor to determine its physical connection to the PIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.



**Figure 10-45. Processor Who Am I Register (WHOAMI)**

[Table 10-49](#) describes the WHOAMI fields.

**Table 10-49. WHOAMI Field Descriptions**

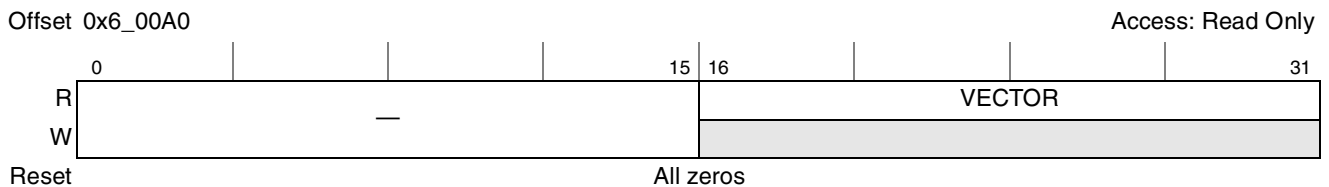
Bits	Name	Description
0–26	—	Reserved
27–31	ID	Returns the ID of the processor reading this register. Always returns zero.

### 10.3.8.4 Processor Interrupt Acknowledge Register (IACK)

In systems based on processors that implement the Power Architecture technology, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK), shown in [Figure 10-46](#). Reading the interrupt acknowledge register returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the interrupt pending register is cleared for edge-sensitive interrupts.
- The in-service register (ISR) is updated.
- The interrupt signal ( $\overline{int}$ ) to the processor is negated.

Reading this register when there is no pending interrupt returns the spurious vector value as described in [Section 10.3.1.8, “Spurious Vector Register \(SVR\).”](#)



**Figure 10-46. Processor Interrupt Acknowledge Register (IACK)**

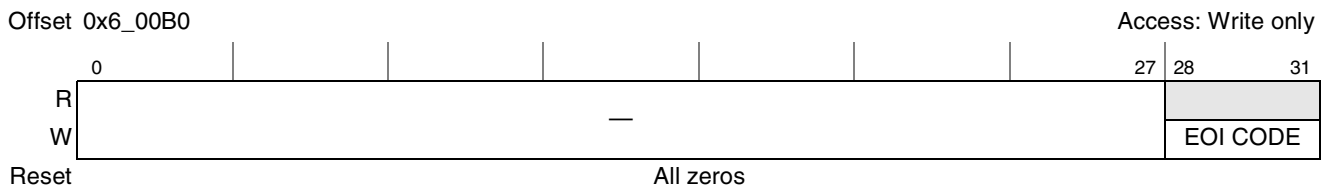
Table 10-50 describes the IACK fields.

**Table 10-50. IACK Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only)

### 10.3.8.5 Processor End of Interrupt Register (EOI)

Writing to the end of interrupt (EOI) register shown in Figure 10-47 signals the end of processing for the highest-priority interrupt currently in-service by the processor. The write to the EOI updates the ISR by retiring the highest priority interrupt. Data values written to this register are ignored, and zero is assumed.



**Figure 10-47. End of Interrupt Register (EOI)**

Table 10-51 describes the EOI fields.

**Table 10-51. EOI Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	EOI CODE	0000 (write only)

## 10.4 Functional Description

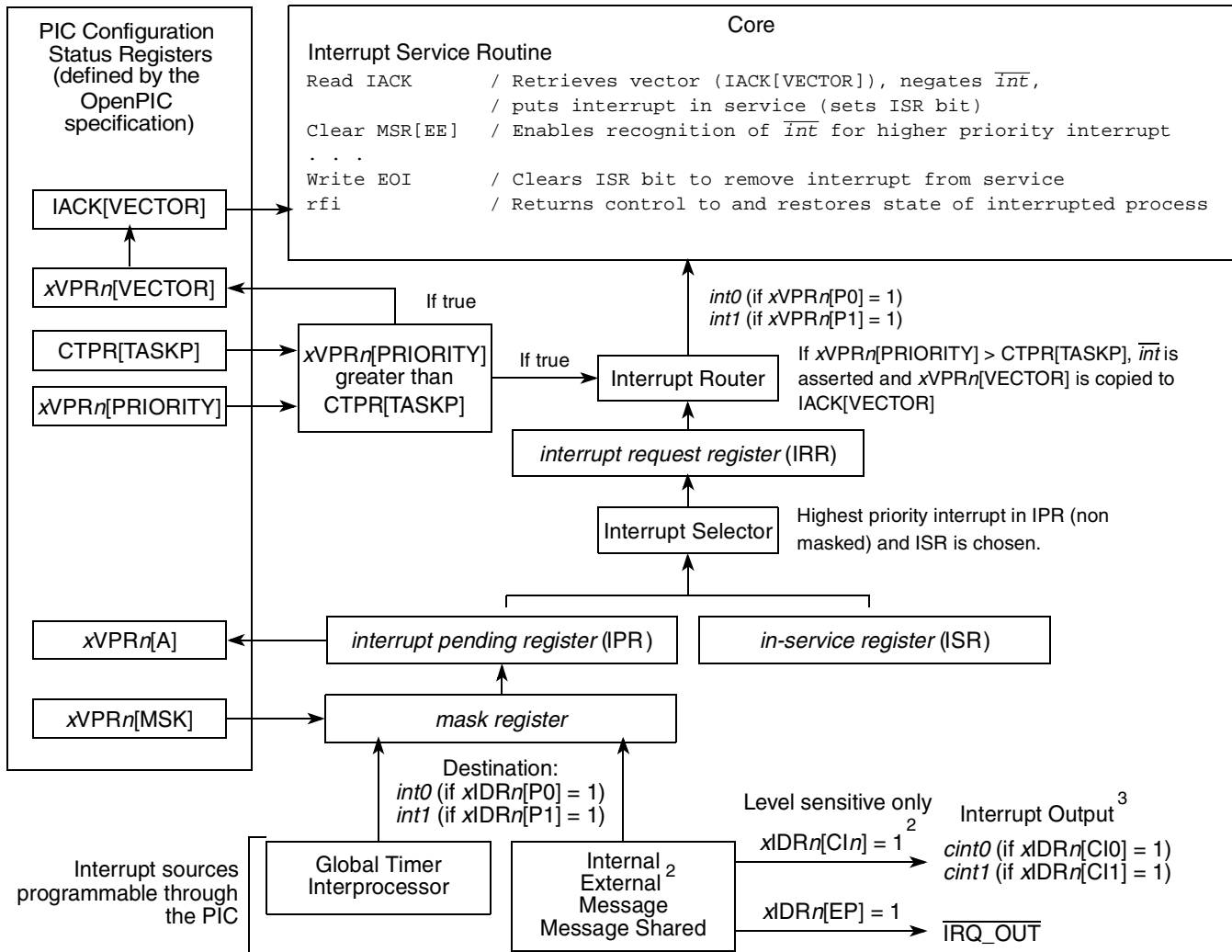
This section is a functional description of the PIC.

### 10.4.1 Flow of Interrupt Control

Figure 10-48 is a block diagram of the PIC unit showing the flow of interrupt processing. This figure is intended to aid in understanding and does not fully represent all internal circuitry of the actual implementation. The PIC receives interrupt signals from both external and internal sources. These signals are qualified and latched in the interrupt pending register (IPR). The IPR feeds the interrupt selector (IS). The interrupt router of the PIC monitors the outputs of its internal interrupt request register (IRR) and other configuration registers. When the priority of the interrupt latched in the IRR is higher than the value in the



processor’s task priority register, the interrupt router asserts the internal interrupt signal ( $\overline{int}$ ) to indicate an interrupt request to the processor. This causes the processor to vector to the external interrupt handler.



<sup>1</sup> If  $\overline{cint}$  or  $\overline{IRQ\_OUT}$  is the destination, EIVPRn[S] must be set to configure the source as level sensitive.  
<sup>2</sup> If multiple destination register bits are set, PIC behavior is undefined.  
<sup>3</sup> Although setting Cln directs the interrupt request to the critical interrupt output ( $\overline{cint0}/\overline{cint1}$ ), integrated logic may connect this signal to a different interrupt input to the core.

**Figure 10-48. PIC Interrupt Processing Flow Diagram**

The interrupt handler executing on the processor should then acknowledge the interrupt by explicitly reading IACK (at which point the interrupt is considered to be in-service). The PIC unit interprets this read as an interrupt acknowledge (IACK) cycle; in response, the PIC unit returns the vector associated with the interrupt source to the interrupt handler routine. The handler can further vector to different branches of interrupt handling accordingly.

Note that reading IACK also negates the interrupt signal to the processor. See Section 10.3.8.4, “Processor Interrupt Acknowledge Register (IACK),” for more details.

The internal in-service register (ISR) tracks all in-service interrupts. An interrupt is considered in-service from the time its vector is read (through an IACK cycle) until the end of interrupt (EOI) register is written, generating what the PIC considers an EOI signal.

### 10.4.1.1 Interrupt Source Priority

Each interrupt source is assigned a priority value through its corresponding vector/priority register. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the interrupt source is greater than the current task priority. Therefore setting a source priority to zero inhibits that interrupt.

The PIC services simultaneous interrupts occurring with the same priority according to the following order:

1. MSG0–MSG3
2. MSI0–MSI7
3. IPI0–IPI3
4. Timer 0–timer 3
5. IRQ[0:11]/PCI INT<sub>x</sub>
6. Internal 0–internal 31

For example, if MSG0, MSG2, and IPI0 all have the same priority and receive simultaneous interrupts, they are serviced in the following order:

1. MSG0
2. MSG2
3. IPI0

### 10.4.1.2 Processor Current Task Priority

The CTPR is set by system software to indicate the relative importance of the task running on the processor. The processor does not receive interrupts with a priority level equal to or lower than its current task priority. Therefore setting the current task priority to 15 for a particular processor prevents the delivery of any interrupt to the processor.

### 10.4.1.3 Interrupt Acknowledge

The PIC unit notifies the processor core of an interrupt by asserting the  $\overline{int}$  signal. When the processor core acknowledges the interrupt request by reading the interrupt acknowledge register (IACK) in the PIC unit, the PIC returns the 16-bit vector associated with the interrupt source to the processor. The interrupt is then considered to be in-service, and remains in-service until the processor performs a write to the PIC unit end of interrupt register (EOI). Writing to the EOI is referred to as an EOI cycle.

## 10.4.2 Nesting of Interrupts

A processor servicing an interrupt, can be interrupted again only if the PIC receives an interrupt request from a source with higher priority than the one being serviced. This is true even if software, as part of its interrupt service routine, writes a new and lower value into the CTPR.

Thus, although several interrupts may be in-service simultaneously, the code currently executing is always handling the highest priority of all the interrupts that are in service. When the processor performs an EOI cycle, this highest priority interrupt is taken out-of-service. The next EOI cycle takes the next-highest priority interrupt out-of-service, and so on. An interrupt with lower priority than those currently in-service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

## 10.4.3 Spurious Vector Generation

Under certain circumstances, the PIC has no valid vector to return to the processor during an interrupt acknowledge cycle. In these cases, the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch:

- $\overline{int}$  is asserted in response to an externally sourced interrupt which is activated with level-sensitive logic, and the asserted level is negated before the interrupt is acknowledged.
- $\overline{int}$  is asserted for an interrupt source that is later masked (using the mask bit in the vector/priority register corresponding to that source) before the interrupt is acknowledged.
- $\overline{int}$  is asserted for an interrupt source that is later masked by an increase in the task priority level before the interrupt is acknowledged.
- An interrupt acknowledge cycle is performed by the processor in spite of the fact that the  $\overline{int}$  signal has not been asserted by the PIC.

In all cases, a spurious vector is not returned if there is another pending interrupt that has sufficient priority to interrupt the processor. If such an interrupt is available, the vector for that interrupt source is returned. The EOI register should not be written in response to the spurious vector. Otherwise, a previously-accepted interrupt might be cleared unintentionally.

## 10.4.4 Messaging Interrupts

There are four 32-bit message registers that can be used to send 32-bit messages to the processor. A messaging interrupt is generated by writing a message register if the corresponding enable bit in the message enable/status register is set, and the interrupt is not masked. Reading the message register or writing a 1 to the status bit clears the interrupt.

## 10.4.5 Shared Message Signaled Interrupts

There are eight shared MSIRs, described in [Section 10.3.6.1, “Shared Message Signaled Interrupt Registers \(MSIRs\)”](#), that indicate which of the interrupt sources sharing the MSI register have pending interrupts. Up to 32 sources can share any individual MSI register. A shared message signaled interrupt is generated by writing to Shared Message Signaled Interrupt Index Register (MSIIR) fields SRS and IBS. This register is primarily intended to support inbound PCI Express message signaled interrupts (MSIs) when the PCI Express controller is configured as a root complex (RC).

MSIIR[SRS] selects the associated MSIR and MSIIR[IBS] selects the interrupt flag/bit in that register that is to be set. The corresponding interrupt needs to be unmasked for the interrupt to occur. A read to an MSIR clears the all of its flags.

### 10.4.6 PCI Express INTx

Whenever the PCI Express controller is in root complex mode and it receives an inbound INTx asserted or negated message transaction, it asserts or negates an equivalent internal INTx signal to the PIC. This INTx virtual-wire interrupt signaling mechanism replaces the PCI standard sideband interrupts (INTA, INTB, INTC, and INTD) that historically were connected to the IRQn external interrupt inputs. The internal INTx signals from the PCI Express controller are logically combined with the interrupt request (IRQn) signals so that they share the same OpenPIC external interrupt controlled by the associated EIVPRn and EIDRn registers.

Table 10-52 details the association of INTx signals to IRQn signals.

**Table 10-52. PCI Express INTx/IRQn Sharing**

PCI Express Number	INTx	IRQn
PCI Express 1	INTA	IRQ0
	INTB	IRQ1
	INTC	IRQ2
	INTD	IRQ3
PCI Express 2	INTA	IRQ4
	INTB	IRQ5
	INTC	IRQ6
	INTD	IRQ7
PCI Express 3	INTA	IRQ8
	INTB	IRQ9
	INTC	IRQ10
	INTD	IRQ11

In general, these signals should be considered mutually exclusive. In particular, if an IRQn signal is being used as active high (EIVPRn[P] = 1), or edge triggered (EIVPRn[S] = 0), the system must not allow inbound PCI Express INTx transactions.

If a PCI Express INTx signal is being used, the PIC must be configured so that external interrupts are active low (EIVPRn[P] = 0) and level sensitive (EIVPRn[S] = 1). In this case, the associated IRQn should be pulled high. Note that it is possible to share IRQn and INTx if the external interrupt is also active-low and level sensitive. However, if an interrupt occurs, the interrupt service routine must poll both the external sources connected to the IRQn input and the PCI Express INTx sources to determine from which path the external interrupt came.

### 10.4.7 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the four internal timers of the PIC unit. The timers can be individually programmed to generate a processor interrupt when they

count down to zero and can be used to generate regular periodic interrupts. Each timer has the following four configuration and control registers:

- Global timer current count register (GTCCR $n$ )
- Global timer base count register (GTBCR $n$ )
- Global timer vector-priority register (GTVPR $n$ )
- Global timer destination register (GTDR $n$ )

The timer frequency should be written to the TFRR. (All of the timers operate at this frequency.) Refer to [Section 10.3.2.1, “Timer Frequency Reporting Register \(TFRR\),”](#) for a description of this register.

Timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in-service, the subsequent interrupt is lost.

The timer control register (TCR) provides users with the ability to create timers larger than the 31-bit global timers. The option also exists to change the timer frequency by setting the appropriate fields of the TCR. See [Section 10.3.2.6, “Timer Control Register \(TCR\).”](#)

## 10.4.8 Reset of the PIC

The PIC unit is reset by a device power-on reset (POR) or by software that sets the GCR[RST] bit. Both of these actions cause the following:

- All pending and in-service interrupts are cleared.
- All interrupt mask bits are set.
- Polarity, sense, external pin, critical interrupt, and activity fields are reset to default values.
- PIR, TFRR, TCR, MER, MSR, and MSGR0–3 are cleared.
- MSG and timer destination fields are set.
- The IPI dispatch registers are cleared.
- All timer base count values are reset to zero and count inhibited.
- The CTPR[TASKP] is reset to 0xF, thus disabling interrupt delivery to the processor.
- The spurious interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.
- The PIC defaults to the pass-through mode (GCR[M] = 0).
- All other registers remain at their pre-reset programmed values.

The GCR[RST] bit is automatically cleared when the reset sequence is complete.

## 10.5 Initialization/Application Information

This section contains initialization and application information for the PIC.

### 10.5.1 Programming Guidelines

The following sections contain information about programming PIC registers.

### 10.5.1.1 PIC Registers

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- IPI dispatch registers and the EOI register, which return zeros on reads.
- Activity bit (A) of the vector/priority registers, which returns the value according to the status of the current interrupt source.
- IACK register, which returns the vector of highest priority which is currently pending, or the spurious vector.
- Reserved fields always return 0.

The following guidelines are recommended when the PIC unit is programmed in mixed mode (GCR[M] = 1):

- All PIC registers must be located in a cache-inhibited and guarded area (through the processor MMU).
- The PIC portion of the address map must be set-up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.
2. Clear the CTPR (CTPR = 0x0000\_0000).
3. Program the PIC to mixed mode by setting GCR[M].
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
  - Load counter with FPR[NIRQ].
  - While counter > 0, perform IACK and EOIs to guarantee all the interrupt pending and in-service registers are cleared.
6. Set the processor CTPR value to the desired value.
7. Set MER as desired (for example, 0x0000\_000F enables all message interrupts). See [Section 10.3.5.2, "Message Enable Register \(MER\),"](#) for more information.

Depending on the interrupt system configuration, the PIC may generate spurious interrupts to clear interrupts latched during power-up. A spurious or non-spurious vector is returned for an interrupt acknowledge cycle in this case. See the programming note below for the non-spurious case.

#### NOTE:

Because the default polarity/sense for external interrupts is edge-sensitive, and edge-sensitive interrupts are not cleared until they are acknowledged, it is possible for the PIC to store spurious edges detected during power-up as pending external interrupts. If software permanently configures an external interrupt source to be edge-sensitive, it may receive the vector for the interrupt source and not a spurious interrupt vector when software clears the mask bit. This can occur once for any edge-sensitive interrupt when its mask bit is first cleared and the PIC is in mixed mode.

To avoid having to handle a false interrupt for this case, software can clear the PIC interrupt pending register of these spurious edge detections by first configuring the polarity/sense of external interrupt sources to be level-sensitive; high-level if the input is a positive-edge source, low-level if it's a negative-edge source (while the mask bit remains set). After this is complete, configuring the external interrupt source as edge-sensitive will not cause a false interrupt.

### 10.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense or destination of an active (unmasked) interrupt source, the following sequence should be performed:

1. Mask the source using the mask (MSK) bit in the vector/priority register.
2. Wait for the activity (A) bit for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.





# Chapter 11

## I<sup>2</sup>C Interfaces

This chapter describes the two inter-IC (IIC or I<sup>2</sup>C) bus interfaces implemented on this device.

### 11.1 Introduction

The I<sup>2</sup>C bus is a two-wire—serial data (SDA) and serial clock (SCL)—bidirectional serial bus that provides a simple efficient method of data exchange between this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. Figure 11-1 shows a block diagram of the two I<sup>2</sup>C interfaces.

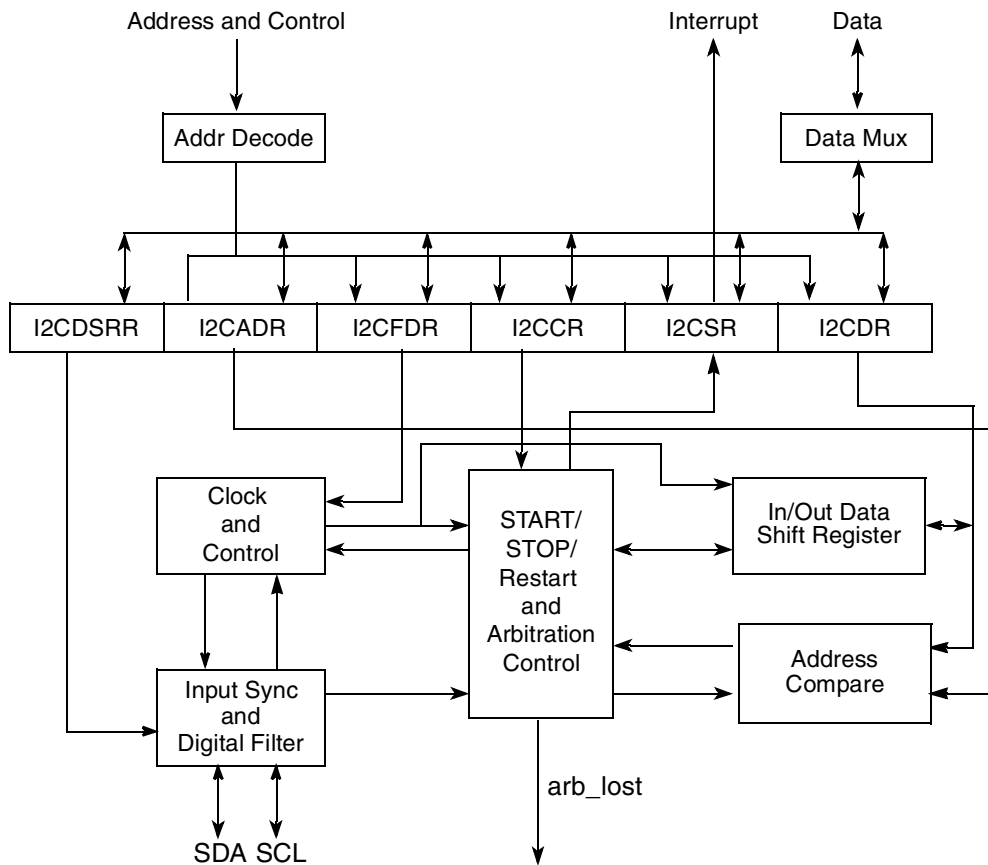


Figure 11-1. I<sup>2</sup>C Block Diagram

### 11.1.1 Overview

The two-wire I<sup>2</sup>C bus minimizes interconnections between devices. The synchronous, multiple-master I<sup>2</sup>C bus allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

### 11.1.2 Features

The I<sup>2</sup>C interface includes the following features:

- Two-wire interface
- Multiple-master operation
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

### 11.1.3 Modes of Operation

The I<sup>2</sup>C units on this device can operate in one of the following modes:

- Master mode—The I<sup>2</sup>C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I<sup>2</sup>C cannot be a master and a slave simultaneously.
- Slave mode—The I<sup>2</sup>C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I<sup>2</sup>C master is detected.
- Interrupt-driven byte-to-byte data transfer—When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/ $\overline{W}$  bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode—This mode can be used to initialize the configuration registers in the device after the I<sup>2</sup>C1 module is initialized. Note that the device powers up with boot sequencer mode disabled as a default, but this mode can be selected with the `cfg_boot_seq[0:1]` power-on reset (POR) configuration signals that are located on the LGPL3 and LGPL5 signals.

Additionally, the following three I<sup>2</sup>C-specific states are defined for the I<sup>2</sup>C interface:

- **START condition**—This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.
- **Repeated START condition**—A START condition that is generated without a STOP condition to terminate the previous transfer.
- **STOP condition**—The master can terminate the transfer by generating a STOP condition to free the bus.

## 11.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

### 11.2.1 Signal Overview

The I<sup>2</sup>C interface uses the SDA and SCL signals, described in [Table 11-1](#), for data transfer. Note that the signal patterns driven on SDA represent address, data, or read/write information at different stages of the protocol.

**Table 11-1. I<sup>2</sup>C Interface Signal Descriptions**

Signal Name	Idle State	I/O	State Meaning
Serial Clock (IICn_SCL)	HIGH	I	When the I <sup>2</sup> C module is idle or acts as a slave, SCL defaults as an input. The unit uses SCL to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.
		O	As a master, the I <sup>2</sup> C module drives SCL along with SDA when transmitting. As a slave, the I <sup>2</sup> C module drives SCL low for data pacing.
Serial Data (IICn_SDA)	HIGH	I	When the I <sup>2</sup> C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I <sup>2</sup> C devices on SDA. The bus is assumed to be busy when SDA is detected low.
		O	When writing as a master or slave, the I <sup>2</sup> C module drives data on SDA synchronous to SCL.

### 11.2.2 Detailed Signal Descriptions

SDA and SCL, described in [Table 11-2](#), serve as a communication interconnect with other devices. All devices connected to these two signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the *MPC8533E Integrated Processor Hardware Specifications* for the electrical characteristics of these signals.

**Table 11-2. I<sup>2</sup>C Interface Signal—Detailed Signal Descriptions**

Signal	I/O	Description
IICn_SCL	I/O	Serial clock. Performs as an input when the device is programmed as an I <sup>2</sup> C slave. SCL also performs as an output when the device is programmed as an I <sup>2</sup> C master.
	O	As outputs for the bidirectional serial clock, these signals operate as described below.
		<b>State Meaning</b>
	I	As inputs for the bidirectional serial clock, these signals operate as described below.
<b>State Meaning</b>		Asserted/Negated—The I <sup>2</sup> C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.
IICn_SDA	I/O	Serial data. Performs as an input when the device is in a receiving mode. SDA also performs as an output signal when the device is transmitting (as an I <sup>2</sup> C master or a slave).
	O	As outputs for the bidirectional serial data, these signals operate as described below.
		<b>State Meaning</b>
	I	As inputs for the bidirectional serial data, these signals operate as described below.
<b>State Meaning</b>		Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

### 11.3 Memory Map/Register Definition

Table 11-3 lists the I<sup>2</sup>C-specific registers and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 11-3. The offsets to the memory map table are defined for both I<sup>2</sup>C interfaces. That is, I<sup>2</sup>C1 starts at address offset 0x000, and I<sup>2</sup>C2 starts at address offset 0x100. The registers for I<sup>2</sup>C1 are listed in Table 11-3, but the registers for I<sup>2</sup>C2 are not. Note that the registers are the same for I<sup>2</sup>C2 except that the offsets change from 0x0nn to 0x1nn.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

All I<sup>2</sup>C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

**Table 11-3. I<sup>2</sup>C Memory Map**

Offset	I <sup>2</sup> C Register	Access	Reset	Section/Page
<b>I<sup>2</sup>C1 Registers</b>				
<b>Block Base Address: 0x0_3000</b>				
0x000	I2CADR—I <sup>2</sup> C address register	R/W	0x00	<a href="#">11.3.1.1/11-6</a>
0x004	I2CFDR—I <sup>2</sup> C frequency divider register	R/W	0x2C	<a href="#">11.3.1.2/11-6</a>
0x008	I2CCR—I <sup>2</sup> C control register	Mixed	0x00	<a href="#">11.3.1.3/11-7</a>
0x00C	I2CSR—I <sup>2</sup> C status register	Mixed	0x81	<a href="#">11.3.1.4/11-9</a>
0x010	I2CDR—I <sup>2</sup> C data register	R/W	0x00	<a href="#">11.3.1.5/11-10</a>
0x014	I2CDFSRR—I <sup>2</sup> C digital filter sampling rate register	R/W	0x10	<a href="#">11.3.1.6/11-11</a>
<b>I<sup>2</sup>C2 Registers</b>				
<b>Block Base Address: 0x0_3000</b>				
0x100– 0x114	I <sup>2</sup> C2 Registers <sup>1</sup>			

<sup>1</sup> I<sup>2</sup>C2 has the same memory-mapped registers that are described for I<sup>2</sup>C1 from 0x000 to 0x014, except the offsets range from 0x100 to 0x114.

### 11.3.1 Register Descriptions

This section describes the I<sup>2</sup>C registers in detail.

#### NOTE

Reserved bits should always be written with the value they returned when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero.

This note does not apply to the I<sup>2</sup>C data register (I2CDR).

### 11.3.1.1 I<sup>2</sup>C Address Register (I2CADR)

Figure 11-2 shows the I2CADR register, which contains the address to which the I<sup>2</sup>C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I<sup>2</sup>C module is in master mode.



Figure 11-2. I<sup>2</sup>C Address Register (I2CADR)

Table 11-4 describes the fields of I2CADR.

Table 11-4. I2CADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I <sup>2</sup> C interface. Note that the default mode of the I <sup>2</sup> C interface is slave mode for an address match. Note that an address match is one of the conditions that can cause I2CSR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved

### 11.3.1.2 I<sup>2</sup>C Frequency Divider Register (I2CFDR)

Figure 11-3 shows the bits of the I<sup>2</sup>C frequency divider register. Refer to application note AN2919, *Determining the I<sup>2</sup>C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

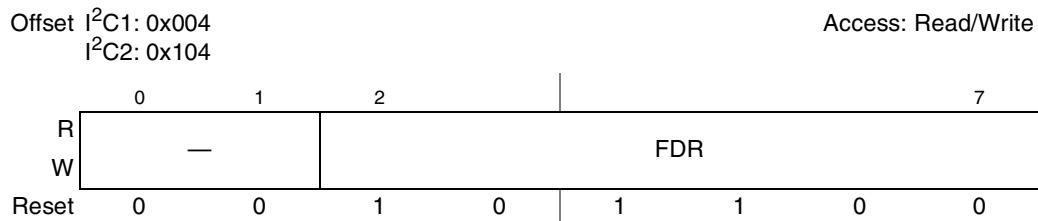


Figure 11-3. I<sup>2</sup>C Frequency Divider Register (I2CFDR)

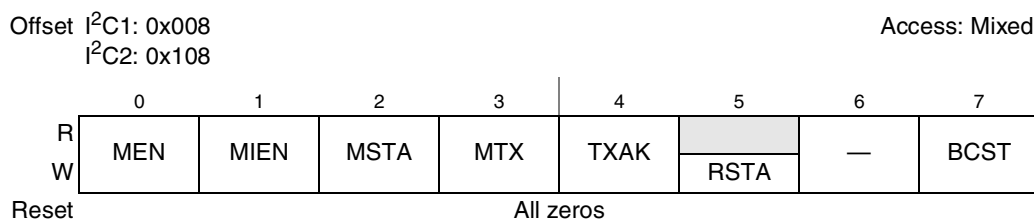
Table 11-5 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

**Table 11-5. I2CFDR Field Descriptions**

Bits	Name	Description																																																																																																																																										
0–1	—	Reserved																																																																																																																																										
2–7	FDR	<p>Frequency divider ratio. Used to prescale the clock for bit rate selection. The serial bit clock frequency of SCL is equal to one third the platform (CCB) clock divided by the designated divider. Note that the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described as follows:</p> <table border="1"> <thead> <tr> <th>FDR</th> <th>Divider (Decimal)</th> <th>FDR</th> <th>Divider (Decimal)</th> <th>FDR</th> <th>Divider (Decimal)</th> </tr> </thead> <tbody> <tr><td>0x00</td><td>384</td><td>0x16</td><td>12288</td><td>0x2B</td><td>1024</td></tr> <tr><td>0x01</td><td>416</td><td>0x17</td><td>15360</td><td>0x2C</td><td>1280 (default)</td></tr> <tr><td>0x02</td><td>480</td><td>0x18</td><td>18432</td><td>0x2D</td><td>1536</td></tr> <tr><td>0x03</td><td>576</td><td>0x19</td><td>20480</td><td>0x2E</td><td>1792</td></tr> <tr><td>0x04</td><td>640</td><td>0x1A</td><td>24576</td><td>0x2F</td><td>2048</td></tr> <tr><td>0x05</td><td>704</td><td>0x1B</td><td>30720</td><td>0x30</td><td>2560</td></tr> <tr><td>0x06</td><td>832</td><td>0x1C</td><td>36864</td><td>0x31</td><td>3072</td></tr> <tr><td>0x07</td><td>1024</td><td>0x1D</td><td>40960</td><td>0x32</td><td>3584</td></tr> <tr><td>0x08</td><td>1152</td><td>0x1E</td><td>49152</td><td>0x33</td><td>4096</td></tr> <tr><td>0x09</td><td>1280</td><td>0x1F</td><td>61440</td><td>0x34</td><td>5120</td></tr> <tr><td>0x0A</td><td>1536</td><td>0x20</td><td>256</td><td>0x35</td><td>6144</td></tr> <tr><td>0x0B</td><td>1920</td><td>0x21</td><td>288</td><td>0x36</td><td>7168</td></tr> <tr><td>0x0C</td><td>2304</td><td>0x22</td><td>320</td><td>0x37</td><td>8192</td></tr> <tr><td>0x0D</td><td>2560</td><td>0x23</td><td>352</td><td>0x38</td><td>10240</td></tr> <tr><td>0x0E</td><td>3072</td><td>0x24</td><td>384</td><td>0x39</td><td>12288</td></tr> <tr><td>0x0F</td><td>3840</td><td>0x25</td><td>448</td><td>0x3A</td><td>14336</td></tr> <tr><td>0x10</td><td>4608</td><td>0x26</td><td>512</td><td>0x3B</td><td>16384</td></tr> <tr><td>0x11</td><td>5120</td><td>0x27</td><td>576</td><td>0x3C</td><td>20480</td></tr> <tr><td>0x12</td><td>6144</td><td>0x28</td><td>640</td><td>0x3D</td><td>24576</td></tr> <tr><td>0x13</td><td>7680</td><td>0x29</td><td>768</td><td>0x3E</td><td>28672</td></tr> <tr><td>0x14</td><td>9216</td><td>0x2A</td><td>896</td><td>0x3F</td><td>32768</td></tr> <tr><td>0x15</td><td>10240</td><td></td><td></td><td></td><td></td></tr> </tbody> </table>	FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)	0x00	384	0x16	12288	0x2B	1024	0x01	416	0x17	15360	0x2C	1280 (default)	0x02	480	0x18	18432	0x2D	1536	0x03	576	0x19	20480	0x2E	1792	0x04	640	0x1A	24576	0x2F	2048	0x05	704	0x1B	30720	0x30	2560	0x06	832	0x1C	36864	0x31	3072	0x07	1024	0x1D	40960	0x32	3584	0x08	1152	0x1E	49152	0x33	4096	0x09	1280	0x1F	61440	0x34	5120	0x0A	1536	0x20	256	0x35	6144	0x0B	1920	0x21	288	0x36	7168	0x0C	2304	0x22	320	0x37	8192	0x0D	2560	0x23	352	0x38	10240	0x0E	3072	0x24	384	0x39	12288	0x0F	3840	0x25	448	0x3A	14336	0x10	4608	0x26	512	0x3B	16384	0x11	5120	0x27	576	0x3C	20480	0x12	6144	0x28	640	0x3D	24576	0x13	7680	0x29	768	0x3E	28672	0x14	9216	0x2A	896	0x3F	32768	0x15	10240				
FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)																																																																																																																																							
0x00	384	0x16	12288	0x2B	1024																																																																																																																																							
0x01	416	0x17	15360	0x2C	1280 (default)																																																																																																																																							
0x02	480	0x18	18432	0x2D	1536																																																																																																																																							
0x03	576	0x19	20480	0x2E	1792																																																																																																																																							
0x04	640	0x1A	24576	0x2F	2048																																																																																																																																							
0x05	704	0x1B	30720	0x30	2560																																																																																																																																							
0x06	832	0x1C	36864	0x31	3072																																																																																																																																							
0x07	1024	0x1D	40960	0x32	3584																																																																																																																																							
0x08	1152	0x1E	49152	0x33	4096																																																																																																																																							
0x09	1280	0x1F	61440	0x34	5120																																																																																																																																							
0x0A	1536	0x20	256	0x35	6144																																																																																																																																							
0x0B	1920	0x21	288	0x36	7168																																																																																																																																							
0x0C	2304	0x22	320	0x37	8192																																																																																																																																							
0x0D	2560	0x23	352	0x38	10240																																																																																																																																							
0x0E	3072	0x24	384	0x39	12288																																																																																																																																							
0x0F	3840	0x25	448	0x3A	14336																																																																																																																																							
0x10	4608	0x26	512	0x3B	16384																																																																																																																																							
0x11	5120	0x27	576	0x3C	20480																																																																																																																																							
0x12	6144	0x28	640	0x3D	24576																																																																																																																																							
0x13	7680	0x29	768	0x3E	28672																																																																																																																																							
0x14	9216	0x2A	896	0x3F	32768																																																																																																																																							
0x15	10240																																																																																																																																											

### 11.3.1.3 I<sup>2</sup>C Control Register (I2CCR)

Figure 11-4 shows the I<sup>2</sup>C control register, I2CCR.



**Figure 11-4. I<sup>2</sup>C Control Register (I2CCR)**

Table 11-6 describes the bit settings of the I2CCR.

**Table 11-6. I2CCR Field Descriptions**

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I <sup>2</sup> C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I <sup>2</sup> C module is enabled. This bit must be set before any other control register bits have any effect. All I <sup>2</sup> C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I <sup>2</sup> C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I <sup>2</sup> C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from one to zero, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from zero to one, a START condition is generated on the bus, and master mode is selected.
3	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I <sup>2</sup> C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeated START. Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration. Note that this bit is not readable, which means if a read is performed to I2CCR[RSTA], a zero value will be returned. 0 No START condition is generated 1 Generates repeated START condition
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I <sup>2</sup> C to accept broadcast messages at address zero



### 11.3.1.4 I<sup>2</sup>C Status Register (I2CSR)

The I<sup>2</sup>C status register, shown in Figure 11-5, is read only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

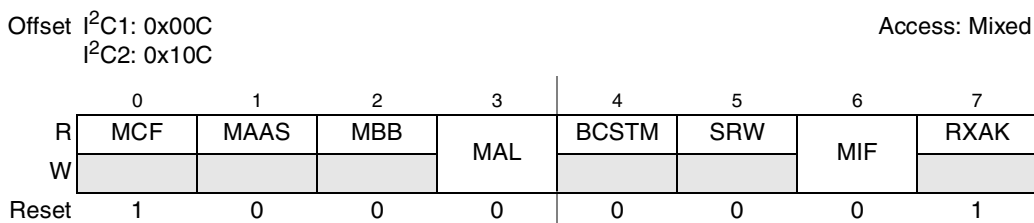


Figure 11-5. I<sup>2</sup>C Status Register (I2CSR)

Table 11-7 describes the bit settings of the I2CSR.

Table 11-7. I2CSR Field Descriptions

Bits	Name	Description
0	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under the following conditions: •When I2CDR is read in receive mode or •When I2CDR is written in transmit mode 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CDR matches with the calling address, this bit is set. The processor is interrupted, if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 0 Not addressed as a slave 1 Addressed as a slave
2	MBB	Bus busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 I <sup>2</sup> C bus is idle 1 I <sup>2</sup> C bus is busy
3	MAL	Arbitration lost. Automatically set when the arbitration procedure is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software 1 Arbitration is lost
4	BCSTM	Broadcast match 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. This will also be set if this I <sup>2</sup> C drives an address of all 0s and broadcast mode is enabled.
5	SRW	Slave read/write. When MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave. This bit is valid only when both of the following conditions are true: •A complete transfer occurred and no other transfers have been initiated. •The I <sup>2</sup> C interface is configured as a slave and has an address match. By checking this bit, the processor can select slave transmit/receive mode according to the command of the master.

Table 11-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] is set). The interrupts for I <sup>2</sup> C1 and I <sup>2</sup> C2 are combined into one interrupt, which is sourced by the dual I <sup>2</sup> C controller. 0 No interrupt is pending. Can be cleared only by software. 1 Interrupt is pending. MIF is set when one of the following events occurs: <ul style="list-style-type: none"> <li>•One byte of data is transferred (set at the falling edge of the 9th clock).</li> <li>•The value in I2CADR matches with the calling address in slave-receive mode.</li> <li>•Arbitration is lost.</li> </ul>
7	RXAK	Received acknowledge. The value of SDA during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

### 11.3.1.5 I<sup>2</sup>C Data Register (I2CDR)

The I2C data register is shown in [Figure 11-6](#).

Figure 11-6. I<sup>2</sup>C Data Register (I2CDR)

[Table 11-8](#) shows the bit descriptions for I2CDR.

Table 11-8. I2CDR Field Descriptions

Bits	Name	Description
0–7	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I <sup>2</sup> C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In master receive mode, reading the data register allows the read to occur, but also allows the I <sup>2</sup> C module to receive the next byte of data on the I <sup>2</sup> C interface. In slave mode, the same function is available after it is addressed. Note that the very first read is always a dummy read.

### 11.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 11-7. Refer to application note AN2919, *Determining the I<sup>2</sup>C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.



Figure 11-7. I<sup>2</sup>C Digital Filter Sampling Rate Register (I2CDFSRR)

Table 11-9 shows the field descriptions for I2CDFSRR.

Table 11-9. I2CDFSRR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I <sup>2</sup> C bus. The resulting sampling rate is calculated by dividing one third the platform (CCB clock) frequency by the non-zero value of DFSR.

## 11.4 Functional Description

The I<sup>2</sup>C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. After the boot sequencer has completed (when powered up in boot sequencer mode), the I<sup>2</sup>C interface will perform as a slave receiver.

Note that the boot sequencer only functions from the I<sup>2</sup>C1 interface; the I<sup>2</sup>C2 interface cannot be used for this purpose.

### 11.4.1 Transaction Protocol

A standard I<sup>2</sup>C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 11-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I<sup>2</sup>C protocol. The details of the protocol are described in the following sections.

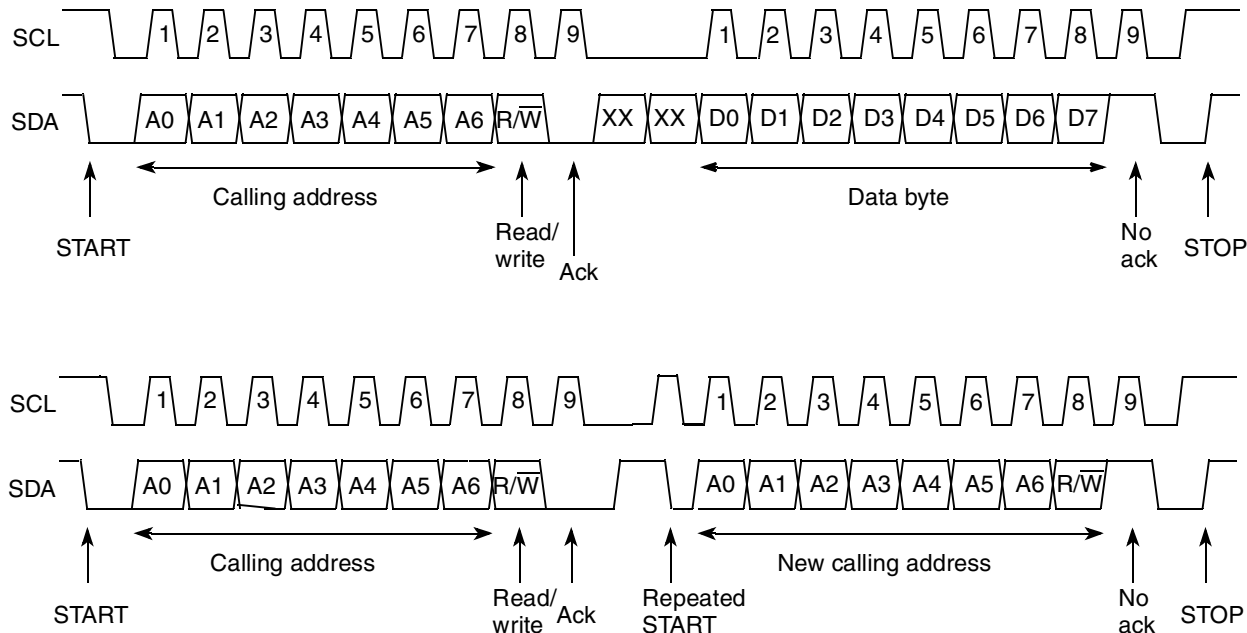


Figure 11-8. I<sup>2</sup>C Interface Transaction Protocol

#### 11.4.1.1 START Condition

When the I<sup>2</sup>C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 11-8, a START condition is defined as a high-to-low transition of SDA while SCL is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets I2CCR[MSTA].

#### 11.4.1.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/ $\overline{W}$  bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, when the I<sup>2</sup>C module is operating as a master, it must not transmit an address that is the same as its slave address. An I<sup>2</sup>C device cannot be master and slave at the same time; if this is attempted, the results are boundedly undefined.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 11-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/ $\overline{W}$  bit sent by the calling master.

The I<sup>2</sup>C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero; however the I<sup>2</sup>C module will not check the R/ $\bar{W}$  bit. The second byte of the broadcast message is the master address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in [Figure 11-8](#). There is one clock pulse on SCL for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the SDA line low at the 9th clock. Therefore, one complete data byte transfer takes 9 clock pulses. Several bytes can be transferred during a data transfer session.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

### 11.4.1.3 Repeated START Condition

[Figure 11-8](#) shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

### 11.4.1.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA signal while SCL is high. For more information, see [Figure 11-8](#). Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2CCR[MSTA].

As described in [Section 11.4.1.3, “Repeated START Condition,”](#) the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

### 11.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in this I<sup>2</sup>C module.

#### 11.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I<sup>2</sup>C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when an SDA rise occurs while SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

#### 11.4.1.5.2 Control Transfer—Implementation Details

The I<sup>2</sup>C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I<sup>2</sup>C. The SCL output is pulled low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Data bit (transmit)
  - Ack bit (receive)
  - START condition
  - STOP condition
  - Restart condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - Ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end

- Slave mode
  - Address cycle
  - Transmit cycle
  - Ack cycle

#### 11.4.1.6 Address Compare—Implementation Details

Address compare block determines if a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The three performed address comparisons are described as follows:

- Whether a broadcast message has been received, to update the I2CSR
- Whether the module has been addressed as a slave, to update the I2CSR and to generate an interrupt
- If the address transmitted by the current master matches the general broadcast address

### 11.4.2 Arbitration Procedure

The I<sup>2</sup>C interface is a true multiple-master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I<sup>2</sup>C module) determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I<sup>2</sup>C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

If the I<sup>2</sup>C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—The I<sup>2</sup>C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—The I<sup>2</sup>C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately results in the current bus master of the I<sup>2</sup>C interface losing arbitration, after which bus operations return to normal.

#### 11.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2CSR[MAL] is set) under the following conditions:

- SDA samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA samples low when the master drives high during a data-receive cycle of the acknowledge (Ack) bit (receive).
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.

- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

Note that the I<sup>2</sup>C module does not automatically retry a failed transfer attempt.

### 11.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

### 11.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
  - Transmit slave address after START condition
  - Transmit slave address after restart condition
  - Transmit data
  - Receive data
- Slave mode
  - Transmit data
  - Receive data
  - Receive slave address after START or restart condition

#### 11.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

#### 11.4.4.2 Input Synchronization and Digital Filter

The following sections describes the synchronizing of the input signals, and the filtering of the SCL and SDA lines in detail.



### 11.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

### 11.4.4.2.2 Filtering of SCL and SDA Lines

The SCL and SDA inputs are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

The sampling rate is equal to a binary value stored in the frequency register I2CDFSRR. The duration of the sampling cycle is controlled by a down counter. This allows a software write to the frequency register to control the filtered sampling rate.

### 11.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

## 11.4.5 Boot Sequencer Mode

If boot sequencer mode is selected on POR (by the settings on the `cfg_boot_seq[0:1]` reset configuration signals, as described in [Section 4.4.3.8, “Boot Sequencer Configuration”](#)), the I<sup>2</sup>C1 module communicates with one or more EEPROMs through the I<sup>2</sup>C interface on IIC1\_SCL and IIC1\_SDA. The boot sequencer accesses the I<sup>2</sup>C1 serial ROM device at the interface frequency designated by the default value of the I2CDFR[DFR] field, 0x2C, which corresponds to a divider of 1280. See [Section 11.3.1.2, “I<sup>2</sup>C Frequency Divider Register \(I2CFDR\)”](#) for additional details of the I2CFDR. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device.

If the boot sequencer is enabled for normal I<sup>2</sup>C addressing mode, the I<sup>2</sup>C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101\_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the 8-bit starting address
5. Generate a repeated START
6. Transmit 0xA1 which is the 7-bit calling address (0b101\_0000) with a read command appended (1 as the least significant bit).
7. Receive 256 bytes of data from the EEPROM (unless the CONT bit is cleared in the data structure).
8. Generate a repeated START

9. Transmit 0xA2 which is the 7-bit calling address of the second target (0b101\_0001) with a write command appended (0 as the least significant bit).
10. Transmit 0x00 which is the 8-bit starting address for the second target.
11. Generate a repeated START
12. Transmit 0xA3 which is the 7-bit calling address (0b101\_0001) with a read command appended (1 as the least significant bit).
13. Receive another 256 bytes of data from the second EEPROM (unless the CONT bit is cleared in the data structure).

The sequence repeats with successive targets until the CONT bit in the data structure is cleared and the CRC check is executed. If the last register is not detected (that is, the CONT bit is never cleared) before wrapping back to the first address, an error condition is detected, causing the device to hang and the HRESET\_REQ signal to assert externally. The I<sup>2</sup>C module continues to read from the EEPROM(s) as long as the continue (CONT) bit is set in the EEPROM(s). The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in [Section 11.4.5.1, “EEPROM Calling Address.”](#) There should be no other I<sup>2</sup>C traffic when the boot sequencer is active.

The boot sequencer mode also supports an extension of the standard I<sup>2</sup>C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes, and this extended addressing mode is selectable during POR with a different encoding on the cfg\_boot\_seq[0:1] reset configuration signals. In this mode, only one EEPROM device may be used, and the maximum number of registers is limited by the size of the EEPROM. If the boot sequencer is enabled for extended I<sup>2</sup>C addressing mode, the I<sup>2</sup>C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101\_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the high-order starting address
5. Transmit 0x00 which is the low-order starting address
6. Generate a repeated START
7. Transmit 0xA1 which is the 7-bit calling address (0b101\_0000) with a read command appended (1 as the least significant bit).
8. Receive data continuously from the EEPROM until the CONT bit is cleared and the CRC check is executed. See [Section 11.4.5.2, “EEPROM Data Format,”](#) for more information.

Note that as described in [Section 4.4.3.8, “Boot Sequencer Configuration,”](#) the default value for the cfg\_boot\_seq[0:1] reset configuration pins is 0b11, which corresponds to the I<sup>2</sup>C boot sequencer being disabled at power-up.

### 11.4.5.1 EEPROM Calling Address

The MPC8533E uses 0b101\_0000 for the EEPROM calling address. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. If more EEPROMs are used, they are addressed in sequential order.

### 11.4.5.2 EEPROM Data Format

The I<sup>2</sup>C module expects that a particular data format be used for data in the EEPROM. A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I<sup>2</sup>C module checks to ensure that this preamble is correctly detected before proceeding further. Following the preamble, there should be a series of configuration registers (known as register preloads) programmed into the EEPROM. Each configuration register should be programmed according to a particular format, as shown in [Figure 11-9](#). The first 3 bytes hold the attributes and address offset, as follows. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302.

After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of the transaction. Byte enables should be asserted for any byte that will be written to the configuration register, and they should be asserted contiguously, creating a 1-, 2-, or 4-byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the LSB of data (data[24:31]).

By setting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If CONT is cleared, the first 3 bytes, including ACS, the byte enables, and the address, must also be cleared. Also, the data contains the final cyclic redundancy check (CRC). A CRC-32 algorithm is used to check the integrity of the data. The polynomial used is:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

CRC values are calculated using the above polynomial with a start value of 0xFFFF\_FFFF and an XOR with 0x0000\_0000. The CRC should cover all bytes stored in the EEPROM prior to the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros). If a preamble or CRC fail is detected, the device hangs and the external HRESET\_REQ signal asserts. If there is a preamble fail, the boot sequencer may continue to pull I<sup>2</sup>C pins low until a hard reset occurs.

0	1	4	5	6	7
ACS	BYTE_EN		CONT	ADDR[0-1]	
ADDR[2-9]					
ADDR[10-17]					
DATA[0-7]					
DATA[8-15]					
DATA[16-23]					
DATA[24-31]					

**Figure 11-9. EEPROM Data Format for One Register Preload Command**

Figure 11-10 shows an example of the EEPROM contents, including the preamble, data format, and CRC.

0	1	2	3	4	5	6	7	
1	0	1	0	1	0	1	0	Preamble
0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	
ACS	BYTE_EN		1	ADDR[0-1]				
ADDR[2-9]								
ADDR[10-17]								
DATA[0-7]								
DATA[8-15]								
DATA[16-23]								
DATA[24-31]								
ACS	BYTE_EN		1	ADDR[0-1]				Second Configuration Preload Command
ADDR[2-9]								
ADDR[10-17]								
DATA[0-7]								
DATA[8-15]								
DATA[16-23]								
DATA[24-31]								
.								
.								
.								

**Figure 11-10. EEPROM Contents**

ACS	BYTE_EN				1	ADDR[0–1]		Last Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
CRC[0–7]								Cyclic Redundancy Check
CRC[8–15]								
CRC[16–23]								
CRC[24–31]								

Figure 11-10. EEPROM Contents (continued)

## 11.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I<sup>2</sup>C interface. Figure 11-11 is a recommended flowchart for I<sup>2</sup>C interrupt service routines.

The I<sup>2</sup>C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. This appropriate byte swapping is needed as I<sup>2</sup>C registers are byte registers. Also, an **msync** assembly instruction must be executed after each I<sup>2</sup>C register read/write access to guarantee in-order execution.

The I<sup>2</sup>C controller does not guarantee its recovery from all illegal I<sup>2</sup>C bus activity. In addition, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I<sup>2</sup>C bus hangs. The recovery routine should also handle the case when the status bits returned after an interrupt are not consistent with what was expected due to illegal I<sup>2</sup>C bus protocol behavior.

### 11.5.1 Initialization Sequence

A hard reset initializes all the I<sup>2</sup>C registers to their default states. The following initialization sequence initializes the I<sup>2</sup>C unit:

1. All I<sup>2</sup>C registers must be located in a cache-inhibited page.
2. Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CCB (platform) clock. Note that the platform frequency must first be divided by two; see Section 11.3.1.2, “I<sup>2</sup>C Frequency Divider Register (I2CFDR),” for more details.
3. Update I2CADR to define the slave address for this device.

4. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CCR[MEN] to enable the I<sup>2</sup>C interface.

### 11.5.2 Generation of START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I<sup>2</sup>C system, test the state of I2CSR[MBB] to check whether the serial bus is free (I2CSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
3. Write the slave address being called into I2CDR. The data written to I2CDR[0–6] comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

The scenario above assumes that the I<sup>2</sup>C interrupt bit (I2CSR[MIF]) is cleared. If MIF is set at any time, an I<sup>2</sup>C interrupt is generated (provided interrupt reporting is enabled with I2CCR[MIEN] = 1) so that the I<sup>2</sup>C interrupt handler can handle the interrupt. Note that the interrupts for I<sup>2</sup>C1 and I<sup>2</sup>C2 are combined into one interrupt, which is sourced by the dual I<sup>2</sup>C controller.

### 11.5.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I<sup>2</sup>C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF]
2. Read the contents of the I<sup>2</sup>C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared. See [Section 11.5.8, “Interrupt Service Routine Flowchart.”](#)

When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] must be toggled at this stage. See [Section 11.5.8, “Interrupt Service Routine Flowchart.”](#)

If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I<sup>2</sup>C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed in order to give the I<sup>2</sup>C signals sufficient time to settle.

During slave-mode address cycles (I2CSR[MAAS] is set), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CSR[SRW] is not valid and I2CCR[MTX] must be read to determine the direction of the current transfer. See [Section 11.5.8, “Interrupt Service Routine Flowchart,”](#) for more details.

### 11.5.4 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data (by setting the transmit acknowledge bit (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I<sup>2</sup>C interface, so the last byte will not receive the data acknowledge (because I2CCR[TXAK] is set). For 1-byte transfers, a dummy read should be performed by the interrupt service routine (see [Section 11.5.8, “Interrupt Service Routine Flowchart”](#)). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I<sup>2</sup>C controller automatically generates a STOP if I2CCR[TXAK] is set. Therefore, I2CCR[TXAK] must be set before allowing the I<sup>2</sup>C module to receive the last data byte on the I<sup>2</sup>C bus. Eventually, I2CCR[TXAK] needs to be cleared again for subsequent I<sup>2</sup>C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

### 11.5.5 Generation of Repeated START

At the end of a data transfer, if the master still wants to communicate on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition. This is accomplished by setting I2CCR[RSTA].

### 11.5.6 Generation of SCL When SDA Low

It is sometimes necessary to force the I<sup>2</sup>C module to become the I<sup>2</sup>C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I<sup>2</sup>C devices to be reset. Thus, SDA can be driven low by another I<sup>2</sup>C device while this I<sup>2</sup>C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force this I<sup>2</sup>C module to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I<sup>2</sup>C module and set the master bit by setting I2CCR to 0x20
2. Enable the I<sup>2</sup>C module by setting I2CCR to 0xA0
3. Read the I2CDR
4. Return the I<sup>2</sup>C module to slave mode by setting I2CCR to 0x80

### 11.5.7 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the module addressed as a slave should be tested to check if a calling of its own address has been received. If I2CSR[MAAS] is set, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/ $\bar{W}$  command bit (I2CSR[SRW]). Writing to I2CCR clears MAAS automatically. MAAS is read as set only in the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers clear MAAS. A data transfer can then be initiated by writing to I2CDR for slave transmits or dummy reading from I2CDR in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode.



### 11.5.7.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit (I2CSR[RXAK]) must be tested before sending the next byte of data. The master signals an end-of-data by not acknowledging the data transfer from the slave. When no acknowledge is received (I2CSR[RXAK] is set), the slave transmitter interrupt routine must clear I2CCR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2CDR then releases SCL so that the master can generate a STOP condition. See [Section 11.5.8, “Interrupt Service Routine Flowchart.”](#)

### 11.5.7.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- I2CSR[MAL] is set
- I2CCR[MSTA] is cleared (changing the master to slave mode)
- An interrupt occurs (if enabled) at the falling edge of the 9th clock of this transfer

Thus, the slave interrupt service routine should first test I2CSR[MAL] and software should clear it if it is set. See [Section 11.4.2.1, “Arbitration Control,”](#) for more information.

## 11.5.8 Interrupt Service Routine Flowchart

[Figure 11-11](#) shows an example algorithm for an I<sup>2</sup>C interrupt service routine. Deviation from the flowchart may result in unpredictable I<sup>2</sup>C bus behavior. However, in the slave receive mode (not shown), the interrupt service routine may need to set I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that an **msync** instruction follow each I<sup>2</sup>C register read or write to guarantee in-order instruction execution.



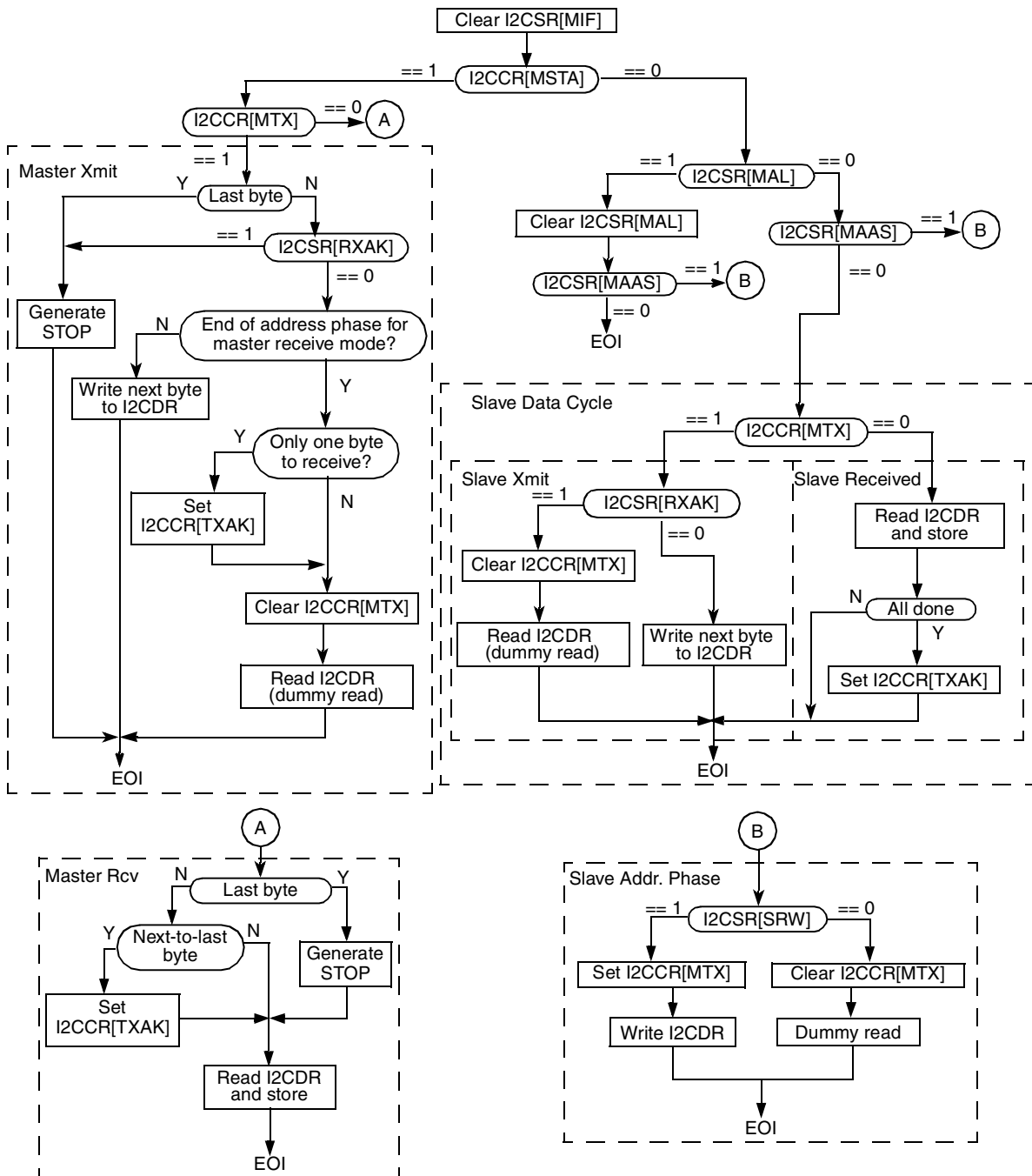


Figure 11-11. Example I<sup>2</sup>C Interrupt Service Routine Flowchart



## Chapter 12

# Security Engine (SEC) 2.1

This chapter describes the functionality of the integrated security engine (SEC 2.1) of the MPC8533E. It addresses the following topics:

- [Section 12.1, “SEC 2.1 Architecture Overview”](#)
- [Section 12.2, “Configuration of Internal Memory Space”](#)
- [Section 12.3, “Descriptor Overview”](#)
- [Section 12.4, “Execution Units”](#)
- [Section 12.5, “Crypto-Channels”](#)
- [Section 12.6, “Security Controller”](#)
- [Section 12.7, “Power-Saving Mode”](#)

The SEC 2.1 is designed to off-load computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the MPC8533E. It is optimized to process all the algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i. The SEC 2.1 is derived from integrated security cores found in other members of the PowerQUICC family, including the SEC 2.0, the version implemented in the MPC8555E and MPC8541E.

The security engine’s execution units (EUs) and primary features include the following:

- PKEU—Public key execution unit that supports the following:
  - RSA (Rivest-Shamir-Adleman) and Diffie-Hellman algorithms
    - Programmable field size up to 2048 bits
  - Elliptic curve cryptography
    - $F_{2^m}$  and  $F(p)$  modes
    - Programmable field size up to 511 bits
- DEU—Data encryption standard execution unit
  - DES, 3DES
  - Two key (K1, K2, K1) or three key (K1, K2, K3)
  - ECB and CBC modes for both DES and 3DES
- AESU—Advanced encryption standard unit
  - Implements the Rijndael symmetric-key cipher
  - ECB, CBC, CCM, and counter modes
  - 128-, 192-, 256-bit key lengths

- AFEU—ARC Four execution unit
  - Implements a stream cipher compatible with the RC4 algorithm
  - 40- to 128-bit programmable key
- KEU—Kasumi execution unit
  - Implements the Kasumi cipher
  - Performs F8 encryption and F9 integrity checking as required for 3GPP
  - Additionally performs A5/3 and GEA-3 modes as used in GSM, EDGE, and GPRS
- MDEU—Message digest execution unit
  - SHA with 160-bit, 224-bit, or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- RNG—Random number generator
- XOR parity generation accelerator for RAID applications
- Four crypto-channels, each supporting a queue of commands (descriptor pointers)
  - Dynamic assignment of crypto-execution units via an integrated controller
  - 256-byte buffer FIFOs on data input and output paths of each execution unit, with flow control for large data sizes
- Master/slave logic, with DMA capability
  - 36-bit address/64-bit data
  - Master interface allows multiple pipelined requests
  - DMA blocks can be on any byte boundary
- Scatter/gather capability
  - Gather capability enables the SEC 2.1 to concatenate multiple segments of memory when reading input data
  - Similarly, scatter capability enables SEC 2.1 to write to multiple segments of memory when writing output data

## 12.1 SEC 2.1 Architecture Overview

The position of the SEC 2.1 (henceforth referred to as SEC) in the MPC8533E architecture is shown in [Figure 12-1](#). The SEC can act as a master on the internal system bus, allowing it to off-load the data movement bottleneck normally associated with slave-only cores. The host processor accesses the SEC through its device drivers using DDR system memory for data storage. The SEC resides in the peripheral memory map of the processor; therefore, when an application requires cryptographic functions, it simply creates descriptors for the SEC which define the cryptographic function to be performed and the location of the data. The SEC's bus-mastering capability permits the host processor to set up a channel with a few short register writes, leaving the SEC to perform reads and writes on system memory to complete the required task.

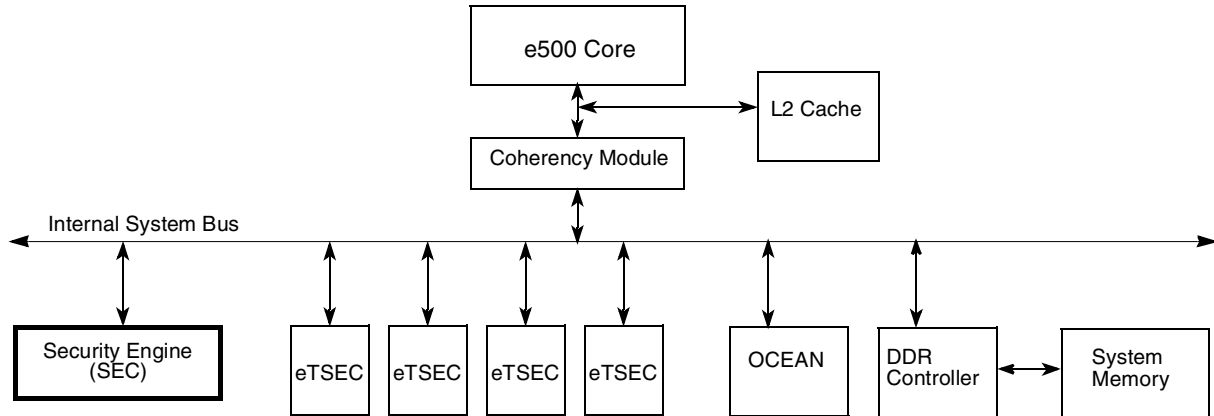


Figure 12-1. SEC Connected to System Bus

A block diagram of the SEC internal architecture is shown in [Figure 12-2](#). The controller block is designed to transfer 64-bit words between the bus and any register inside the SEC.

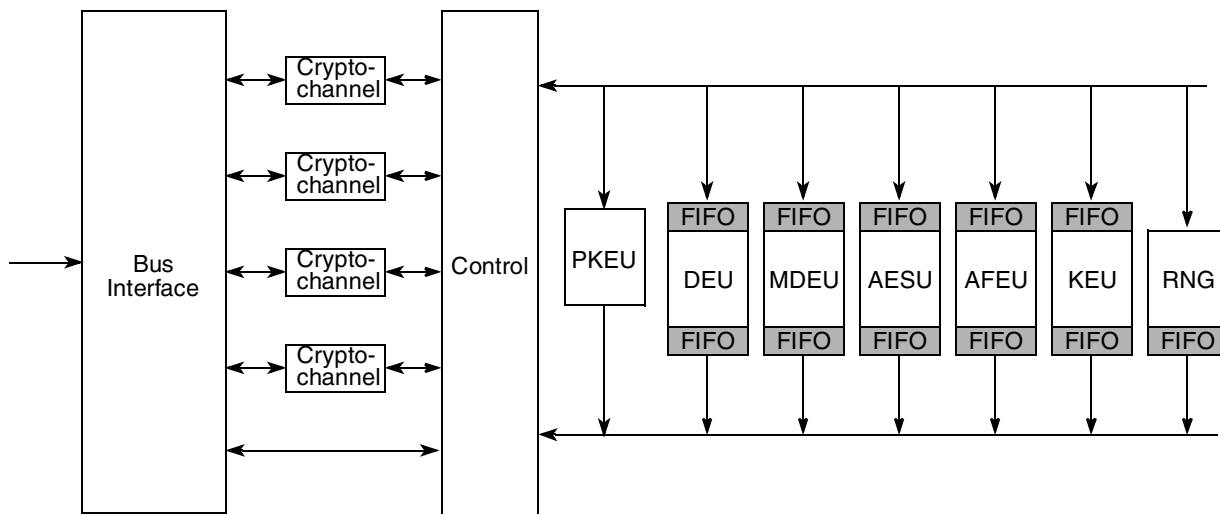


Figure 12-2. SEC Functional Modules

An SEC operation begins when the host writes a descriptor pointer to the fetch FIFO in one of the four SEC crypto-channels. From this point on, the channel directs the sequence of operations. The channel uses the descriptor pointer to read the descriptor, then decodes the first word of the descriptor to determine the operation to be performed and the crypto-execution units needed to perform it. The channel requests the controller to assign the needed crypto-execution units. Next, the channel requests that the controller fetch the keys, context, and data from locations specified in the rest of the descriptor. The controller satisfies the requests by making requests to the master interface per the programmable priority scheme. Data is fed into the execution units through their registers and input FIFOs. The execution units read from their input FIFOs and write processed data to their output FIFOs. The channel requests the controller to write data from the output FIFOs and registers, back to system memory via the master/slave interface.

The channel can signal to the host that it is done with a descriptor via interrupt or by a writeback of the descriptor header into host memory. For more about this signaling, see [Section 12.5, “Crypto-Channels.”](#)

Upon completion of a descriptor, the channel checks the next entry in its fetch FIFO, and, if non-zero, the channel requests a burst read of the next descriptor.

For most packets, the entire payload is too long to fit in an execution unit's input or output FIFO. The SEC then uses a flow control scheme for reading and writing data. The channel directs the controller to read bursts of input as necessary to keep refilling the input FIFO, until the entire payload has been fetched. Similarly, the channel directs the controller to write bursts of output whenever enough accumulates in the execution unit's output FIFO.

## 12.1.1 Descriptors

As a crypto-acceleration block, the SEC controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through descriptors. A descriptor specifies a cryptographic function to be performed, and contains pointers to all necessary input data and to the places where output data is to be written. Some descriptor types perform multiple functions to facilitate particular protocols. A sample descriptor is shown in [Table 12-1](#).

**Table 12-1. Example Descriptor**

Field Name	Value	Description
Header	0x2053_1E08_0000_0000	Example header for IPsec ESP outbound using DES and MD-5
Length0 Extent0 Pointer0	16 0 (32 or 36-bit pointer)	Number of bytes in authenticate key Unused Pointer to authentication key
Length1 Extent1 Pointer1	16 0 (32 or 36-bit pointer)	Number of bytes in authentication-only data Unused Pointer to authentication-only data
Length2 Extent2 Pointer2	8 0 (32 or 36-bit pointer)	Length of input context (IV) Unused Pointer to input context
Length3 Extent3 Pointer3	8 0 (32 or 36-bit pointer)	Number of bytes in cipher key Unused Pointer to cipher key
Length4 Extent4 Pointer4	1500 0 (32 or 36-bit pointer)	Number of bytes of data to be ciphered Unused Pointer to input data to perform ciphering upon
Length5 Extent5 Pointer5	1500 12 (32 or 36-bit pointer)	Number of bytes of data after ciphering Number of bytes in authentication result (ICV) Pointer to location where cipher output is to be written, followed by ICV
Length6 Extent6 Pointer6	8 0 (32 or 36-bit pointer)	Length of output Context (IV) Unused Pointer to location where altered Context is to be written

Each descriptor contains eight long-words (64 bits each), consisting of the following:

- One long-word of header—The header describes the required services and encodes information that indicates which EUs to use and which modes to set. It also indicates whether notification should be sent to the host when the descriptor operation is complete.

- Seven long-words containing pointers and lengths used to locate input or output data. Each pointer can either point directly to the data, or can point to a link table that lists a set of data segments to be concatenated.

For more information, refer to [Section 12.3, “Descriptor Overview.”](#)

## 12.1.2 Execution Units (EUs)

Execution unit (EU) is the generic term for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The EUs are compatible with IPsec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE 802.11i processing, and can work together to perform high-level cryptographic tasks. The SEC’s execution units are as follows:

- PKEU for computing asymmetric key operations, including modular exponentiation (and other modular arithmetic functions) or ECC point arithmetic
- DEU for performing block cipher, symmetric key cryptography using DES and 3DES
- AFEU for performing RC-4 compatible stream cipher symmetric key cryptography
- AESU for performing the Advanced Encryption Standard algorithm and XOR acceleration
- MDEU for performing security hashing using MD-5, SHA-1, or SHA-256
- KEU for performing 3GPP confidentiality and integrity (F8 and F9) algorithms.
- RNG for random number generation

Each EU is described in detail in [Section 12.4, “Execution Units.”](#)

### 12.1.2.1 Public Key Execution Unit (PKEU)

The PKEU is capable of performing many advanced mathematical functions to support both RSA and ECC public-key cryptographic algorithms. ECC is supported in both  $F(2)^m$  (polynomial-basis) and  $F(p)$  modes. This EU supports all levels of functions to assist the host microprocessor to perform its desired cryptographic function. For example, at the highest level, the accelerator performs modular exponentiations to support RSA and performs point multiplies to support ECC. At the lower levels, the PKEU can perform simple operations such as modular multiplies. For more information, refer to [Section 12.4.1, “Public Key Execution Unit \(PKEU\).”](#)

#### 12.1.2.1.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 160 bits and 512 bits in programmable increments of 8, with each programmable value  $i$  supporting all actual field sizes from  $8i - 7$  to  $8i$ . The result is hardware supporting a wide range of cryptographic security. Larger field/modulus sizes result in greater security but lower performance; processing time is determined by field or modulus size. For example, a field size of 160 is roughly equivalent to the security provided by 1024-bit RSA. A field size set to 208 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations which are addition, multiplication, inverse, and squaring. Point add and

double each use of all four finite field operations. Similarly, point multiplication uses all elliptic curve point operations as well as the finite field operations. All these functions are supported both in modular arithmetic as well as polynomial basis finite fields.

### 12.1.2.1.2 Modular Exponentiation Operations

The PKEU is also capable of performing ordinary integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures (including ECDSA) and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the SEC's PKEU include the following (refer to [Table 12-10](#) for a complete list):

- $R^2 \bmod N$
- $A^E \bmod N$
- $(A \times B) R^{-1} \bmod N$
- $(A \times B) R^{-2} \bmod N$
- $(A + B) \bmod N$
- $(A - B) \bmod N$

where  $A' = AR \bmod N$ ,  $N$  is the modulus vector,  $A$  and  $B$  are input vectors,  $E$  is the exponent vector, and  $R$  is  $2^s$ , where  $s$  is the bit length of the  $N$  vector rounded up to the nearest multiple of 32.

The PKEU can perform modular arithmetic on operands up to 2048 bits in length. The modulus must be larger than or equal to 97 bits (13 bytes). This is not seen as a limitation since for sizes smaller than this, no useful cryptographic application exists. Furthermore, for small data sizes the overhead of using a hardware accelerator would not be justified. The PKEU uses the Montgomery modular multiplication algorithm to perform core functions. The addition and subtraction functions help support known methods of the Chinese remainder theorem (CRT) for efficient implementation of the RSA algorithm.

### 12.1.2.2 Data Encryption Standard Execution Unit (DEU)

The DES execution unit (DEU) performs bulk data encryption/decryption, in compliance with the Data Encryption Standard algorithm (ANSI x3.92). The DEU can also compute 3DES, an extension of the DES algorithm in which each 64-bit input block is processed three times. The SEC supports 2-key ( $K1=K3$ ) or 3-key 3DES.

The DEU operates by permuting 64-bit data blocks with a shared 56-bit key and an initialization vector (IV). The SEC supports two modes of operation: electronic code book (ECB) and cipher clock chaining (CBC).

For more information, refer to [Section 12.4.2, "Data Encryption Standard Execution Unit \(DEU\)."](#)

### 12.1.2.3 ARC Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning a byte of plain text is encrypted with a key to produce a byte of ciphertext. The key is variable length and the AFEU supports key lengths from 8 to



128 bits (in byte increments), providing a wide range of security strengths. ARC4 is a symmetric algorithm, meaning each of the two communicating parties share the same key.

For more information, refer to [Section 12.4.3, “ARC Four Execution Unit \(AFEU\).”](#)

#### 12.1.2.4 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using either the MD5, SHA-1, SHA-225 or SHA-256 algorithms for bulk data hashing. With any hash algorithm, the larger message is mapped onto a smaller output space; therefore collisions are possible, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
- SHA-224 and SHA-256 are cryptographic hash functions that provide integrity protection against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

For more information, refer to [Section 12.4.4, “Message Digest Execution Unit \(MDEU\).”](#)

#### 12.1.2.5 Random Number Generator (RNG)

The RNG is a functional block capable of generating 64-bit random numbers. It is designed to comply with FIPS 140-1 standards for randomness and non-determinism.

Because many cryptographic algorithms use random numbers as a source for generating a secret value (a nonce), it is desirable to have a private RNG for use by the SEC. The anonymity of each random number must be maintained, as well as the unpredictability of the next random number. The FIPS-140 common-criteria compliant private RNG allows the system to develop random challenges or random secret keys. The secret key can thus remain hidden from even the high-level application code, providing an added measure of physical security.

For more information, refer to [Section 12.4.5, “Random Number Generator \(RNG\).”](#)

#### 12.1.2.6 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the Advanced Encryption Standard Rijndael algorithm. The AESU executes on 128 bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AESA is a symmetric-key algorithm, the sender and receiver use the same key for both encryption and decryption. The session key and initialization vector (IV) are supplied to the AESU module prior to encryption. The processor supplies data to the module that is processed as 128 bit input. The AESU operates in ECB, CBC, CTR, and CCM modes.

The AESU is also used for performing the exclusive OR (XOR) operation used to generate parity data for RAID storage applications. When operating in this mode, no session keys are involved, and the AESU XORs up to three data streams at a time to produce parity data.

For more information, refer to [Section 12.4.6, “Advanced Encryption Standard Execution Unit \(AESU\).”](#)

### 12.1.2.7 Kasumi Execution Unit (KEU)

The KEU is a functional block capable of encrypting/decrypting and/or performing integrity checks on 64-bit blocks of data using a 128-bit key. The KEU is designed to be compatible with both the F8 and F9 algorithms as defined in the ETSI/SAGE specification document 1 for the 3GPP standard. In addition, if the KEU is supplied with an original MAC value, it is capable of performing a bitwise check of this original MAC against an F9 MAC generated by the KEU.

The KEU also performs Kasumi-based algorithms that are used to protect data in 2G and 2.5G protocols, specifically A5/3 for GSM and EDGE, and GEA-3 for GPRS.

For more information, refer to [Section 12.4.7, “Kasumi Execution Unit \(KEU\).”](#)

## 12.1.3 Crypto-Channels

The SEC includes four crypto-channels that manage data and EU function. Each channel contains the following:

- A fetch FIFO, which holds a queue of pointers to descriptors waiting to be serviced
- A configuration register, which allows the user a number of options for SEC event signaling
- Control registers containing information about the transaction in process
- A status register containing an indication of the last unfulfilled bus request
- A descriptor buffer memory used to store the active descriptor
- Scatter and gather link table buffer memory used to store the active link table

Whenever a crypto-channel is idle and its fetch FIFO is non-empty, the channel reads the next descriptor pointer from the fetch FIFO. Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. To service this descriptor, the channel directs execution of the following steps:

1. Analyze the descriptor header to determine the cryptographic services required, and request use of the appropriate EUs from the controller.
2. Wait for the controller to grant access to the required EUs.
3. Set the appropriate mode bits in the EU(s) for the required service.
4. Fetch data parcels using pointers from the descriptor buffer, and place them in either an EU input FIFO or EU registers (as appropriate). The term data parcel refers here to any input or output of a cryptographic process, such as a key, hash result, input context, output context, or text-data. Context refers to either an initialization vector (IV) or other internal EU state that can be read out or loaded in. Text-data refers to plaintext or ciphertext to be operated on.
5. If the data size is greater than EU FIFO size, continue fetching input data, and writing output data to memory.
6. Wait for EU(s) to complete processing.

7. Upon completion, unload results from output FIFOs and context registers and write them to external memory using pointers in the descriptor buffer.
8. If multiple services are requested, go back to step 3.
9. Release the EUs.
10. If done notification is enabled, perform this notification.

The channel can signal to the host that it is done with a descriptor via interrupt or by a writeback of the descriptor header into host memory. In the case of writeback, the value written back is identical to the header that was read, with the exception that a DONE field is set to all 1s. In addition, the channel can be configured to write back other status fields that indicate the result of ICV checking (if any). The user can opt to enable this signaling at end of every descriptor, or at the end of selected descriptors. For more information about configuring signaling, see [Section 12.5.1.1, “Crypto-Channel Configuration Registers 1–4 \(CCCRn\).”](#)

Many security protocols involve both encryption and hashing of packet payloads. To accomplish this without requiring two passes through the data, channels can configure data flows through more than one EU. In such cases, one EU is designated the primary EU, and the other as the secondary EU. The primary EU receives its data from memory via the controller, and the secondary EU receives its data by snooping the SEC buses.

There are two types of snooping.

- Input data can be fed to the primary EU and the same input data snooped by the secondary EU. This is called in-snooping.
- Output data from the primary EU can be snooped by the secondary EU. This is called out-snooping.

In the SEC, the secondary EU is always the MDEU.

For more information, refer to [Section 12.5, “Crypto-Channels.”](#)

## 12.1.4 Controller

The controller manages on-chip resources, including the individual execution units (EUs), FIFOs, the master/slave interface to the MPC8533E internal system bus, and the other internal buses that connect the remaining modules. The controller receives service requests from the master/slave interface and from the crypto-channels, and schedules the required activities. The controller provides for two ways of operating the execution units:

- Channel-controlled access—A channel can request a particular service from any available execution unit. This is the normal operating condition.
- Host-controlled access—The host can move data into and out of any execution unit directly through memory-mapped EU registers. This is typically only used for debug.

The system bus interface and access to system memory are critical factors in performance, and the 64-bit master/slave interface of the SEC controller allows it to achieve performance unattainable on secondary buses.

### 12.1.4.1 Channel-Controlled Access

Processing begins when a descriptor pointer is written to the fetch FIFO of one of the channels. Based on the services requested by the descriptor header, the channel asks the controller to assign the necessary EUs to that channel. If all appropriate EUs are already reserved by other channels, the channel stalls and waits to fetch data until an appropriate EU is available. If multiple channels simultaneously request the same EU, the EU is assigned on a weighted priority or round-robin basis.

Once the required EU has been reserved, the channel requests that the controller fetch and load the appropriate data. The controller acts as a master on the system bus, reading and writing on byte boundaries. The channel operates the EU, and makes further requests to the controller to write output data to system memory. When the descriptor processing is complete, the channel asks the controller to release the EU for use by other channels.

### 12.1.4.2 Host-Controlled Access

All execution units (EUs) are memory-mapped, and can be used entirely through register read/write access. The SEC operates as a slave, and the host must write the information typically provided through the descriptor into the appropriate registers and FIFOs of the SEC. This method is more CPU intensive, and requires a great deal of familiarity with the SEC registers. It is recommended that host-controlled access be used only for operations using a single EU, and for debug purposes.

For more information, refer to [Section 12.6, “Security Controller.”](#)

## 12.2 Configuration of Internal Memory Space

[Table 12-2](#) shows the base address map, while [Table 12-3](#) provides the address map, including all registers in the execution units. These address values are offsets from CCSRBAR.

Note that these tables show modulo-8 addresses; the three least-significant address bits that are used to select bytes within 64-bit words are not shown.

**Table 12-2. SEC Base Address Map**

Address Offset (AD 17-0)	Module	Description	Type	Reference
0x3_0000–0x3_0FFF	—	Reserved	—	—
0x3_1000–0x3_10FF	Controller	Arbiter/controller control register space	Resource control	<a href="#">12.6/12-104</a>
0x3_1100–0x3_11FF	Channel_1	Channel 1	Data control	<a href="#">12.5/12-91</a>
0x3_1200–0x3_12FF	Channel_2	Channel 2		
0x3_1300–0x3_13FF	Channel_3	Channel 3		
0x3_1400–0x3_14FF	Channel_4	Channel 4		

Table 12-2. SEC Base Address Map (continued)

Address Offset (AD 17-0)	Module	Description	Type	Reference
0x3_2000-0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	<a href="#">12.4.2/12-33</a>
0x3_4000-0x3_4FFF	AESU	AES execution unit		<a href="#">12.4.6/12-67</a>
0x3_6000-0x3_6FFF	MDEU	Message digest execution unit		<a href="#">12.4.4/12-50</a>
0x3_8000-0x3_8FFF	AFEU	Arc Four execution unit		<a href="#">12.4.3/12-41</a>
0x3_A000-0x3_AFFF	RNG	Random number generator		<a href="#">12.4.5/12-62</a>
0x3_C000-0x3_CFFF	PKEU	Public key execution unit		<a href="#">12.4.1/12-26</a>
0x3_E000-0x3_EFFF	KEU	Kasumi execution unit		<a href="#">12.4.7/12-79</a>

Table 12-3 shows the system address map showing all functional registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 12-3. SEC Address Map

Address Offset (AD 17-0)	Register	Access	Reset	Reference
<b>Controller</b>				
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	<a href="#">12.6.5.2/12-110</a>
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.6.5.3/12-111</a>
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	<a href="#">12.6.5.4/12-112</a>
0x3_1020	ID—Identification register	R	0x0030_0000_0010_0000	<a href="#">12.6.5.5/12-113</a>
0x3_1BF8	IP block revision	R	0x0030_0000_0010_0000	<a href="#">12.6.5.6/12-113</a>
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	<a href="#">12.6.5.1/12-109</a>
0x3_1030	MCR—Master control register	R/W	0000_0000_0000_0000	<a href="#">12.6.5.7/12-114</a>
<b>Channel 1</b>				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000_0000_0000	<a href="#">12.5.1.1/12-92</a>
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	<a href="#">12.5.1.2/12-95</a>
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000_0000_0000	<a href="#">12.5.1.3/12-100</a>

Table 12-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Register	Access	Reset	Reference
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000_0000_0000	<a href="#">12.5.1.4/12-101</a>
0x3_1180–0x3_11BF	DB1—Crypto-channel 1 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	<a href="#">12.5.1.5/12-102</a>
0x3_11C0–0x3_11DF	Gather link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
0x3_11E0–0x3_11FF	Scatter link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
<b>Channel 2</b>				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000_0000_0000	<a href="#">12.5.1.1/12-92</a>
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	<a href="#">12.5.1.2/12-95</a>
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000_0000_0000	<a href="#">12.5.1.3/12-100</a>
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000_0000_0000	<a href="#">12.5.1.4/12-101</a>
0x3_1280–0x3_12BF	DB2—Crypto-channel 2 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	<a href="#">12.5.1.5/12-102</a>
0x3_12C0–0x3_12DF	Gather link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
0x3_12E0–0x3_12FF	Scatter link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
<b>Channel 3</b>				
0x3_1308	CCCR3—Crypto-channel3 configuration register	R/W	0x0000_0000_0000_0000	<a href="#">12.5.1.1/12-92</a>
0x3_1310	CCPSR3—Crypto-channel3 pointer status register	R	0x0000_0000_0000_0007	<a href="#">12.5.1.2/12-95</a>
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000_0000_0000	<a href="#">12.5.1.3/12-100</a>
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000_0000_0000	<a href="#">12.5.1.4/12-101</a>
0x3_1380–0x3_13BF	DB3—Crypto-channel 3 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	<a href="#">12.5.1.5/12-102</a>
0x3_13C0–0x3_13DF	Gather link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
0x3_13E0–0x3_13FF	Scatter link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
<b>Channel 4</b>				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000_0000_0007	<a href="#">12.5.1.1/12-92</a>
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0000	<a href="#">12.5.1.2/12-95</a>
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000_0000_0000	<a href="#">12.5.1.3/12-100</a>
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000_0000_0000	<a href="#">12.5.1.4/12-101</a>

Table 12-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Register	Access	Reset	Reference
0x3_1480–0x3_14BF	DB4—Crypto-channel 4 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	<a href="#">12.5.1.5/12-102</a>
0x3_14C0–0x3_14DF	Gather link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
0x3_14E0–0x3_14FF	Scatter link tables	W	0x0000_0000_0000_0000	<a href="#">12.5.1.6/12-102</a>
<b>Data Encryption Standard Execution Unit (DEU)</b>				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.1/12-33</a>
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.2/12-34</a>
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.3/12-35</a>
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.4/12-35</a>
0x3_2028	DEUSR—DEU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.2.5/12-36</a>
0x3_2030	DEUIR—DEU interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.2.6/12-37</a>
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	<a href="#">12.4.2.7/12-39</a>
0x3_2050	DEUEUG—DEU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.2.8/12-40</a>
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.9/12-41</a>
0x3_2400	DEUK1—DEU key register 1	W	—	<a href="#">12.4.2.10/12-41</a>
0x3_2408	DEUK2—DEU key register 2	W	—	<a href="#">12.4.2.10/12-41</a>
0x3_2410	DEUK3—DEU key register 3	W	—	<a href="#">12.4.2.10/12-41</a>
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000_0000_0000	<a href="#">12.4.2.11/12-41</a>
<b>Advanced Encryption Standard Execution Unit (AESU)</b>				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.1/12-67</a>
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.2/12-69</a>
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.3/12-69</a>
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.4/12-70</a>
0x3_4028	AESUSR—AESU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.6.5/12-71</a>
0x3_4030	AESUIR—AESU interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.6.6/12-72</a>
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.6.7/12-73</a>
0x3_4050	AESUEUG—AESU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.6.8/12-74</a>
0x3_4100–0x3_4108	AESU context memory registers	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.9/12-75</a>
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.9.5/12-79</a>
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000_0000_0000	<a href="#">12.4.6.9.6/12-79</a>
<b>Message Digest Execution Unit (MDEU)</b>				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.4.1/12-50</a>
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.4.3/12-54</a>
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.4.4/12-54</a>
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.4.5/12-55</a>
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.4.6/12-55</a>



Table 12-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Register	Access	Reset	Reference
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.4.7/12-57</a>
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.4.8/12-58</a>
0x3_6040	MDEUICVSR—MDEU ICV size register	W	0x0000_0000_0000_0000	<a href="#">12.4.4.9/12-59</a>
0x3_6050	MDEUEUG—MDEU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.4.10/12-59</a>
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000_0000_0000	<a href="#">12.4.4.11/12-60</a>
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000_0000_0000	<a href="#">12.4.4.12/12-61</a>
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000_0000_0000	<a href="#">12.4.4.13/12-62</a>
<b>ARC Four Execution Unit (AFEU)</b>				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.1/12-42</a>
0x3_808	AFEUKSR—AFEU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.3/12-43</a>
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.4/12-44</a>
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.5/12-44</a>
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.3.6/12-45</a>
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.3.7/12-46</a>
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.3.8/12-48</a>
0x3_8050	AFEUEUG—AFEU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.3.9/12-49</a>
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.10.1/12-49</a>
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.10.2/12-50</a>
0x3_8400	AFEUK1—AFEU key register 0	W	—	<a href="#">12.4.3.11/12-50</a>
0x3_8480	AFEUK2—AFEU key register 1	W	—	<a href="#">12.4.3.11/12-50</a>
0x3_8800–0x3_8FFF (3_8E00)	AFEU FIFO	R/W	0x0000_0000_0000_0000	<a href="#">12.4.3.12/12-50</a>
<b>Random Number Generator (RNG)</b>				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.5.1/12-63</a>
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.5.2/12-63</a>
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.5.3/12-63</a>
0x3_A028	RNGSR—RNG status register	R	0x0000_0000_0000_0000	<a href="#">12.4.5.4/12-64</a>
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.5.5/12-65</a>
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.5.6/12-66</a>
0x3_A050	RNGEUG—RNG EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.5.7/12-67</a>
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000_0000_0000	<a href="#">12.4.5.8/12-67</a>
<b>Public Key Execution Units (PKEU)</b>				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.1/12-26</a>
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.2/12-27</a>
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.4/12-28</a>



Table 12-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Register	Access	Reset	Reference
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.5/12-29</a>
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.1.6/12-29</a>
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000_0000_0000	<a href="#">12.4.1.7/12-30</a>
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.1.8/12-31</a>
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.3/12-28</a>
0x3_C050	PKEUEUG—PKEU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.1.9/12-32</a>
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C340–0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C380–0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C3C0–0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C400–0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
0x3_C800–0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_0000_0000	<a href="#">12.4.1.10/12-32</a>
<b>Kasumi Execution Unit (KEU)</b>				
0x3_E000	KEUMR—KEU mode register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.1/12-80</a>
0x3_E008	KEUKSR—KEU key size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.2/12-81</a>
0x3_E010	KEUDSR—KEU data size register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.3/12-81</a>
0x3_E018	KEURCR—KEU reset control register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.4/12-82</a>
0x3_E028	KEUSR—KEU status register	R	0x0000_0000_0000_0000	<a href="#">12.4.7.5/12-83</a>
0x3_E030	KEUISR—KEU interrupt status register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.6/12-84</a>
0x3_E038	KEUICR—KEU interrupt control register	R/W	0x0000_0000_0000_1000	<a href="#">12.4.7.7/12-85</a>
0x3_E048	KEUDOR—KEU data out register (F9 MAC)	R	0x0000_0000_0000_0000	<a href="#">12.4.7.8/12-87</a>
0x3_E050	KEUEUG—KEU EU go register	W	0x0000_0000_0000_0000	<a href="#">12.4.7.9/12-87</a>
0x3_E100	KEUIV1—KEU initialization vector 1 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.10/12-88</a>
0x3_E108	KEUICV—KEU ICV_In register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.11/12-88</a>
0x3_E110	KEUIV2—KEU initialization vector 2 register (Fresh)	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.12/12-89</a>
0x3_E118	KEUC1—KEU context_1 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>
0x3_E120	KEUC2—KEU context_2 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>
0x3_E128	KEUC3—KEU context_3 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>
0x3_E130	KEUC4—KEU context_4 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>
0x3_E138	KEUC5—KEU context_5 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>

Table 12-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Register	Access	Reset	Reference
0x3_E140	KEUC6—KEU context_6 register	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.13/12-89</a>
0x3_E400	KEUKD1—KEU key data register_1 (CK-high)	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.14/12-89</a>
0x3_E408	KEUKD2—KEU key data register_2 (CK-low)	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.14/12-89</a>
0x3_E410	KEUKD3—KEU key data register_3 (IK-high)	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.15/12-90</a>
0x3_E418	KEUKD4—KEU key data register_4 (IK-low)	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.15/12-90</a>
0x3_E800–0x3_EFFF	KEUFIFO	R/W	0x0000_0000_0000_0000	<a href="#">12.4.7.16/12-90</a>

## 12.3 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host has determined that a security operation is required, it creates a descriptor containing all the information the SEC needs to perform the security operation. The host creates the descriptor in main memory, then writes a pointer to the descriptor into the fetch FIFO of one of the SEC channels. The channel uses this pointer to read the descriptor into its descriptor buffer. Once it obtains the descriptor, the SEC uses its bus mastering capability to obtain inputs and write results, thus off-loading data movement and encryption operations from the host processor.

For test purposes, it is also possible for the host to write keys, context, and text-data directly to execution units, using SEC’s host-controlled access. This method avoids the use of descriptors.

### 12.3.1 Descriptor Structure

SEC descriptors are conceptually similar to descriptors used by most devices with DMA capability. The descriptors have a fixed length of 64 bytes, i.e. eight 64-bit words (referred to as dwords). A descriptor consists of one header dword and seven pointer dwords, as shown in [Figure 12-3](#).

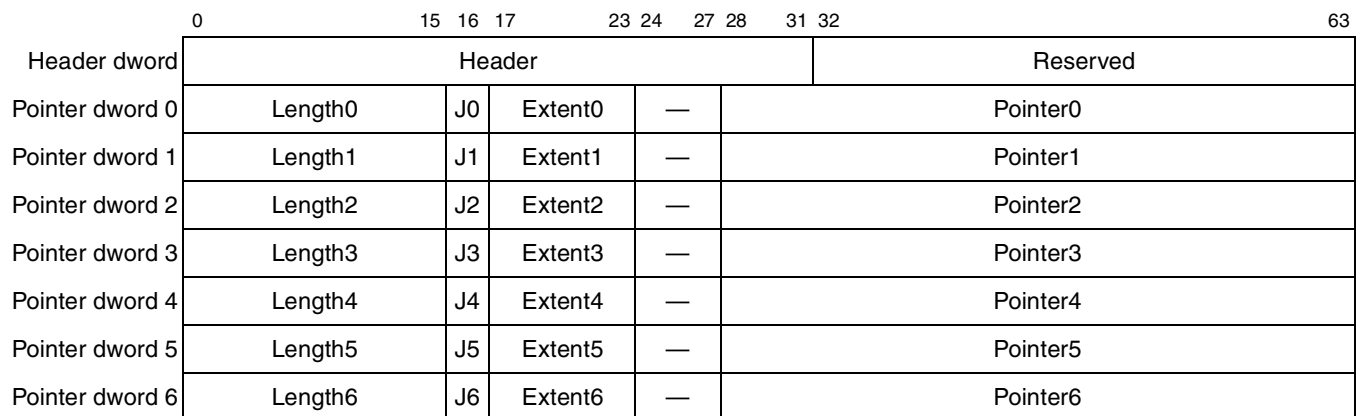


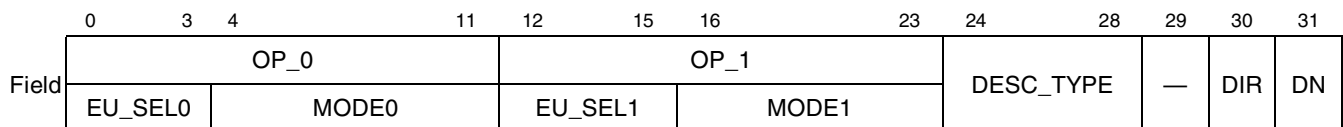
Figure 12-3. Descriptor Format

The header dword specifies the security operation to be performed, the execution unit(s) needed, and the modes for each execution unit. The pointer dwords, all of which have the same format, contain pointer and length information for locating input or output data parcels (such as keys, context, or text-data). The large number of pointers provided in the descriptor allows for multi-algorithm operations that require fetching of multiple keys, as well as fetch and return of contexts. Any pointer dword that is not needed can be given a length of zero, and the channel will skip over the corresponding operations.

SEC descriptors include scatter/gather capability, which means that each pointer in a descriptor can be either a direct pointer to a contiguous parcel of data, or can be a pointer to a link table which is a list of pointers and lengths used to assemble the data parcel. When a link table is used to read input data, this is referred to as a gather operation; when used to write output data, it is referred to as a scatter operation.

### 12.3.2 Descriptor Format: Header Dword

Descriptors are created by the host to guide the SEC through required cryptographic operations. The header dword defines the operations to be performed, the mode for each operation, and internal addressing used by the controller and channel for internal data movement. The fields that must be supplied to SEC are shown in the field rows of [Figure 12-4](#), and described in [Table 12-4](#). The SEC device drivers allow the host to create proper headers for each cryptographic operation.



**Figure 12-4. Header Dword**

Header dword bit definitions are described below.

**Table 12-4. Header Dword Bit Definitions**

Bits	Name	Description
OP_0		
0–3	EU_SEL0	Primary EU select. See <a href="#">Section 12.3.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,”</a> for possible values.
4–11	MODE0	Primary mode. Mode data used to program the primary EU. The mode data is to the chosen EU. This field is passed directly to bits 56–63 of the mode register in the selected EU.
OP_1		
12–15	EU_SEL1	Secondary EU select. See <a href="#">Section 12.3.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,”</a> for possible values.
16–23	MODE1	Secondary mode. Mode data used to program the primary EU. The mode data is to the chosen EU. This field is passed directly to bits 56–63 of the mode register in the selected EU.
24–28	DESC_TYPE	Descriptor type. This field, along with DIR, determines the sequence of actions to be performed by the channel and selected EUs using the blocks of data listed in the rest of the descriptor. The attributes determined include the direction of data flow for each data block, which EU (primary or secondary) is accessed, what snooping options are used, and which internal EU addresses are accessed. See <a href="#">Section 12.3.2.2, “Selecting Descriptor Type—DESC_TYPE,”</a> for possible values.
29	—	Reserved.

**Table 12-4. Header Dword Bit Definitions (continued)**

Bits	Name	Description
30	DIR	Direction. Direction of overall data flow. 0 Outbound 1 Inbound This, along with the DESC_TYPE field, helps determine the sequence of actions to be performed by the channel and selected EUs.
31	DN	Done notification. 0 No done notification. 1 Signal done to the host on completion of this descriptor. This enables done notification if the NT field is 1 in the channel configuration register (see <a href="#">Table 12-50</a> ). The done notification can take the form of an interrupt, a writeback in the DONE field of this header dword (see <a href="#">Table 12-51</a> ), or both, depending upon the states of the CDIE (channel done interrupt enable) and CDWE (channel done writeback enable) bits in the channel configuration register.

### 12.3.2.1 Selecting Execution Units—EU\_SEL0 and EU\_SEL1

[Table 12-5](#) shows the values for EU\_SEL0 and EU\_SEL1 in the descriptor header. The following rules govern the choices for these fields:

1. EU\_SEL0 values of no EU selected or reserved result in an unrecognized header error condition during processing of the descriptor header.
2. The only valid choices for EU\_SEL1 are No EU selected or MDEU. Any other choice results in an unrecognized header error condition.
3. If EU\_SEL1 is MDEU, then EU\_SEL0 must be DEU, AESU, or AFEU. All other values of EU\_SEL0 result in an unrecognized header error condition.

**Table 12-5. EU\_SEL0 and EU\_SEL1 Values**

Value (binary)	Selected EU
0000	No EU selected
0001	AFEU
0010	DEU
0011	MDEU
0100	RNG
0101	PKEU
0110	AESU
0111	KEU
others	Reserved
1111	Reserved for header writeback

### 12.3.2.2 Selecting Descriptor Type—DESC\_TYPE

Table 12-6 shows the permissible values for the DESC\_TYPE field in the descriptor header. Descriptor types from the SEC1.0, which have 0 in the last bit, are listed first, followed by new SEC 2.0 types, which have a 1 in the last bit.

**Table 12-6. Descriptor Types**

Value (binary)	Descriptor Type	Notes
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnooping <sup>1</sup>
0001_0	common_nonsnoop	Common, nonsnooping, non-PKEU, non-AFEU
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_0	—	Reserved
0111_0	—	Reserved
1000_0	pkeu_mm	PKEU-Montgomery multiplication
1001_0	—	Reserved
1010_0	—	Reserved
1011_0	—	Reserved
1100_0	hmac_snoop_aesu_ctr	AESU CTR hmac snooping <sup>2</sup>
1101_0	—	Reserved
1110_0	—	Reserved
1111_0	—	Reserved
0000_1	ipsec_esp	IPsec ESP mode encryption and hashing
0001_1	802.11i AES_ccmp	CCMP encryption and hashing, suitable for 802.11i
0010_1	srtplib	SRTP encryption and hashing
0011_1	pkeu_assemble	pkeu_assemble elliptic curve cryptography
0100_1	pkeu_ptmul	pkeu_ptmul elliptic curve cryptography
0101_1	pkeu_ptadd_dbl	pkeu_ptadd_dbl elliptic curve cryptography
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher
1001_1	tls_ssl_stream	TLS/SSL generic stream cipher
1010_1	raid_xor	XOR data streams
All others	—	Reserved

<sup>1</sup> Type 0000\_0 is for AES-CTR operations. Type 0001\_0 also supports AES-CTR, however to use AES-CTR with 0001\_0, the user must prepend zeros to the AES-Ctx before loading the AES context registers.

<sup>2</sup> Type 1100\_0 is for AES-CTR operations with HMAC. Type 0010\_0 also supports AES-CTR with HMAC, however to use AES-CTR with 0010\_0, the user must prepend zeros to the AES-Ctx before loading the AES context registers.

For more about descriptor types and the data used for each type, see [Section 12.3.5, “Descriptor Types.”](#)

### 12.3.3 Descriptor Format: Pointer Dwords

The descriptor contains seven pointer dwords which define where in memory the SEC should access its input and output data parcels. The pointer dwords are numbered 0 to 6 as shown in [Figure 12-3](#). The channel determines how it will use each of the pointer dwords based on the DESC\_TYPE and DIR fields in the header. The channel accesses the first data parcel by starting at a location given by a POINTER value, and accessing a number of bytes given by a LENGTH or EXTENT value. Subsequent data parcels may be accessed by starting where a previous data parcel ended, or by starting at a different POINTER. The LENGTH or EXTENT used with any POINTER may be from the same pointer dword or from a different pointer dword in the same descriptor. Although the EXTENT field exists in each pointer dword of the SEC descriptor, only the EXTENTS in pointer dwords 3, 4, and 5 are currently in use.

If the extend address enable field is high (CCR[EAE]; see [Table 12-50](#)), then the four EPTR bits are concatenated with the POINTER field to form a 36-bit pointer address.

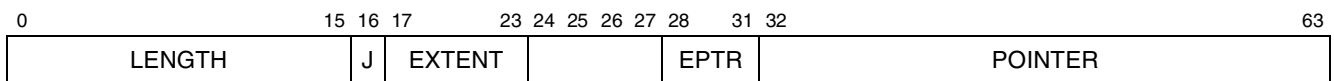


Figure 12-5. Pointer Dword

Table 12-7. Pointer Dword Field Definitions

Bits	Name	Description
0–15	LENGTH	Length. A number of bytes in the range 0 to 65535. The use of this field depends on the DESC_TYPE and DIR fields in the header dword. A value of zero causes the channel to skip this dword.
16	J	Jump. Determines whether to jump to a link table whenever the POINTER field in this same lword is used. 0 The POINTER field points to data. 1 The POINTER field points to a link table, and scatter/gather is enabled.
17–23	EXTENT	Extent. A number of bytes in the range 0 to 127. The use of this field depends on the DESC_TYPE and DIR fields in the header dword.
24–27	—	Reserved.
28–31	EPTR	Extended pointer. Concatenated as the top 4 bits of the pointer when CCR[EAE] is set. See <a href="#">Section 12.5.1.1, “Crypto-Channel Configuration Registers 1–4 (CCCRn)”</a> .
32–63	POINTER	Pointer. A memory address.

On occasion, a descriptor field may not be applicable to the requested service. With seven pointer dwords, it is possible that not all these dwords will be required to specify the input and output parameters. (Some operations, for example, do not require context.) Where a particular dword is not used, all fields should be set to 0.

Some descriptors involve more than seven parcels of input and output data. In these cases, it is necessary to use one POINTER field to address a sequence of data parcels.

LENGTH and EXTENT fields normally specify the sizes of data parcels. In some cases, however, the POINTER field is zero, and the LENGTH and/or EXTENT fields simply specify values to be written to an EU.

The J bit in each pointer dword is used to enable the scatter/gather feature. If a data parcel to be read or written by SEC is in one contiguous block of memory locations, then the scatter/gather feature is not

needed. In this case the POINTER should be set to point directly at the first byte of the parcel, and the J bit should be 0. On the other hand, if the data parcel is stored in several separate segments of memory, then the scatter/gather capability is needed to assemble or distribute the complete parcel. In this case the POINTER should be set to point to a link table, and the J bit should be 1. For link table format, see [Section 12.3.4, “Link Table Format.”](#) Scatter/gather capability is available for all pointer dwords of all descriptor types, with the following exception(s):

- The RAID-XOR descriptor type does not allow scatter/gather.

### 12.3.4 Link Table Format

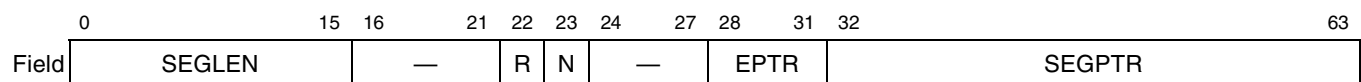
Link tables implement scatter/gather capability. For gather operations, a link table specifies a list of memory segments that are to be concatenated in the process of assembling data parcels. For scatter operations, a link table specifies a list of memory segments into which the output data should be written. Scatter or gather of a data parcel may be specified by a single link table or by a chain of link tables that are linked together with pointers (see [Figure 12-7](#)).

The link table or chain of link tables accessed through some descriptor POINTER must specify enough memory segments to hold all the data that will be accessed through that pointer. In most cases, only a single data parcel is accessed through a given POINTER, and the chain of link tables specifies just that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of data parcels, and the chain of link tables must supply data for the entire sequence. If a link table is used to access a sequence of data parcels, the end of each parcel must also be at the end of a memory segment. In other words, a single memory segment must not straddle two data parcels. An example of proper construction of link tables is illustrated in [Figure 12-7](#).

A link table may contain any number of long word entries. There are two kinds of entries, regular entries and next entries. Each regular entry specifies a memory segment by means of a 36-bit starting address (SEGPTR) and a 16-bit length (SEGLen). A next entry is used at the end of a link table to specify that the list of memory segments is continued in another link table. In a next entry, the N bit is set, the SEGPTR field gives the address of the next link table, and the SEGLen field must be 0. A chain of link tables may contain any number of link tables.

Whether the list of memory segments is in a single link table or split into several link tables, the last entry in the last link table is a regular entry with the R (return) bit set. The R bit signifies the end of link table operations so that the channel returns to the descriptor for its next pointer (if any).

Link tables are illustrated in [Figure 12-7](#). A single link table entry is shown in [Figure 12-6](#).



**Figure 12-6. Link Table Entry**



Table 12-8. Link Table Field Definitions

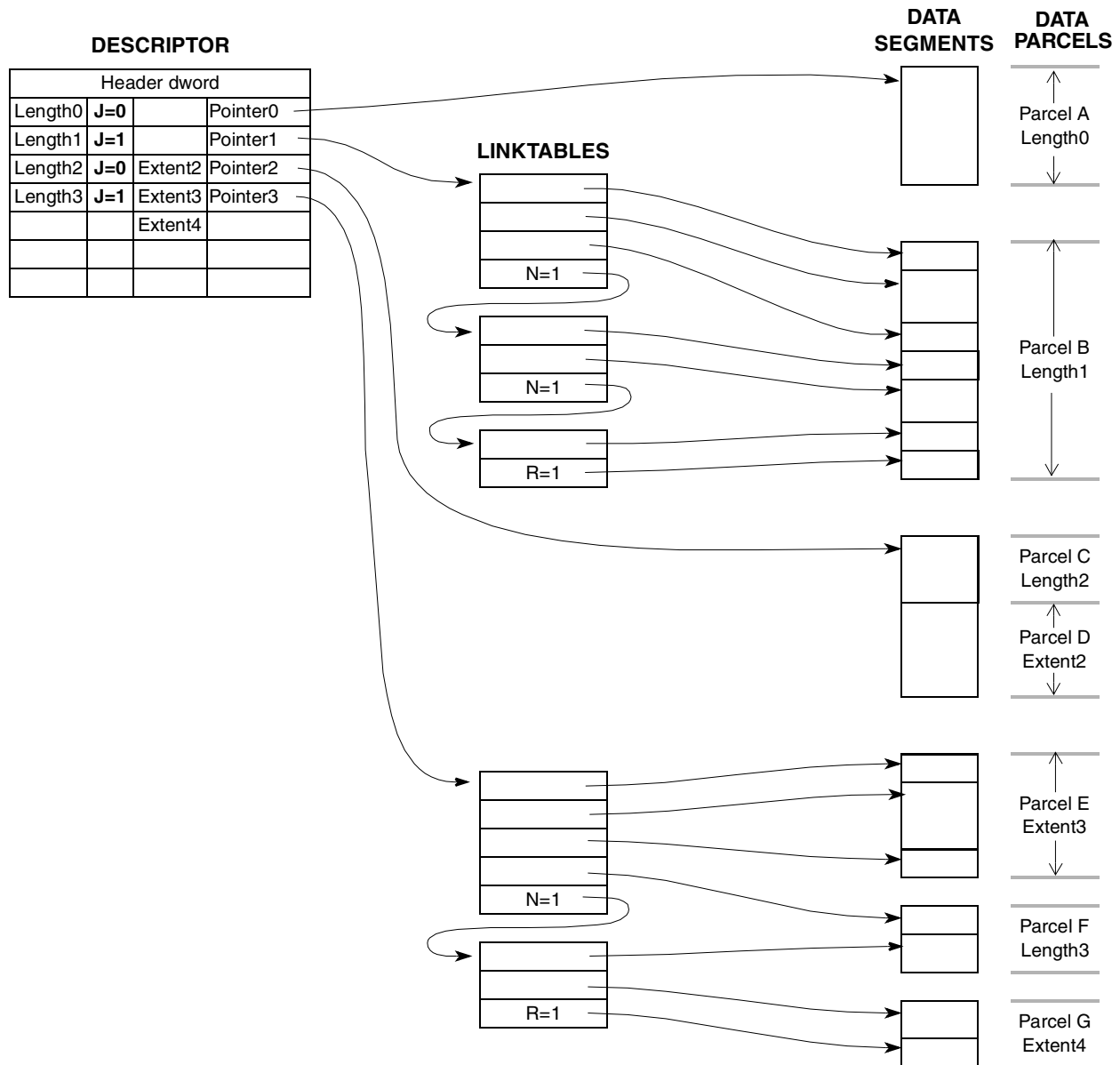
Bits	Name	Description
0–15	SEGLEN	Length. When N=0, a number in the range 1 to 65535, specifying the number of bytes in the memory segment. pointed to by SEGPtr. A value of 0 will cause an error state to be set in the channel pointer status register—G-STATE for a gather operation or S-STATE for a scatter operation (see Section 12.5.1.2). When N=1, must be 0.
16–21	—	Reserved
22	R	Return. When N=0: 0 No special action. 1 This is the last entry in the chain of link tables. If this entry does not specify the right number of bytes to complete the last data parcel, a G-STATE or S-STATE error will be set in the channel pointer status register (see Section 12.5.1.2). When N=1, ignored.
23	N	Next. 0 No special action. 1 This is the last long word in the current link table. The SEGPtr field is the address of the next link table in the chain.
24–27	—	Reserved
28–31	EPTR	Extended Pointer. Concatenated as the top 4 bits of the segment pointer when EAE is high (see the EAE bit in Table 1-51 on page 1-116).
32–63	SEGPtr	Segment pointer: A memory address.

For any sequence of data parcels accessed by a link table or chain of link tables, the combined lengths of the parcels (the sum of their LENGTH and/or EXTENT fields) must equal the combined lengths of the link table memory segments (SEGLEN fields). Otherwise the channel sets the appropriate error state in the channel pointer status register—G-STATE for gather error or S-STATE for scatter error (see Section 12.5.1.2).

Example (from Figure 12-7): To demonstrate use of a link table, assume that the current descriptor type calls for the channel to read a data parcel using Pointer3 and Extent3 fields, and assume that J3 = 1. Due to the J3 value, Pointer3 is not used as a data address but instead used as the address of a link table. The channel begins by reading the first four long words starting at Pointer3 into an internal ‘gather table buffer’.

Using the first entry of the gather table buffer, the channel starts accessing the data parcel by reading SEGLEN bytes beginning at SEGPtr. If the required data parcel size (Extent3) is greater than this first SegLen, the channel moves on to the next entry of the gather table buffer, and reads SEGLEN bytes starting at SEGPtr. While there are more bytes to be read in the data parcel, this process continues. If the channel’s gather table buffer is exhausted, the channel reads the next four long words of the link table into its gather table buffer. If a gather table buffer entry is encountered in which the N bit is set, the channel uses the SEGPtr field in that word to find the next link table in the chain. The last byte of the required parcel size (Extent3) must coincide with the last byte of a memory segment, or unpredictable results may occur.





**Figure 12-7. Descriptors, Link Tables, and Data Parcels**

This figure illustrates various ways that a descriptor may specify data parcels:

The first pointer dword in the descriptor specifies Parcel A using the simplest method—the parcel is specified directly through Pointer0 and Length0.

The next pointer dword uses a chain of link tables to specify Parcel B. Since J=1, Pointer1 is used as the address of a link table. The link table specifies several “regular” entries specifying data segments to be concatenated. The last word of the link table is a “next” entry indicating that the list continues in the next link table. The last entry in the last link table of the chain has the R bit set.

The last cases illustrate how one pointer in a descriptor can be used to specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterwards, with length specified by Extent2. Pointer3 is used for three data parcels (E, F, and G), this time using link tables.

Now assume that the channel accesses its next data parcel using Pointer3 again, this time with length given by Length3. In this case the channel continues to the next line of the link table, and begins reading the memory segment specified there. As before, the channel concatenates memory segments from as many link table entries as necessary to obtain the required number of bytes (Length3).

Similarly, the next data parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 data parcel is the last one to be accessed through Pointer3. Then the link table entry that supplies the last memory segment for Extent4 has the R bit set, signifying that this is the last entry in the chain of link tables.

## 12.3.5 Descriptor Types

Table 12-9 shows how the pointer dwords should be used with the various descriptor types to load keys, context, and text-data into the execution units, and how the required outputs should be unloaded.

Additional explanation of the use of certain descriptor types, and the meaning of the pointer dwords can be found in the *SEC 2.1 Descriptor Programmer's Guide*.

**Table 12-9. Descriptor Format by Type**

Descriptor Type	field type	Pointer Dword1	Pointer Dword2	Pointer Dword 3	Pointer Dword4	Pointer Dword 5	Pointer Dword 6	Pointer Dword 7
0000_0 aesu_ctr_ nosnoop	Length	nil	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out	nil
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0001_0 common_ nosnoop	Length	nil	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out	nil
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0010_0 hmac_snoop_ no_afha	Length	HMAC Key	HMAC Data	Cipher Key	Cipher IV	In FIFO	Out FIFO	HMAC Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0101_0 common_ nosnoop_ afha	Length	nil	ARC-4 Context (In FIFO)	ARC-4 Key	In FIFO	Out FIFO	ARC-4 Context (Out FIFO)	nil
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
1000_0  pkmm	Length	N	B	A	E	B Out	nil	nil
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
1100_0 hmac_snoop_ aesu_ctr	Length	HMAC Key	HMAC Data	AES Key	AES Ctx	In FIFO	Out FIFO	HMAC Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0000_1  ipsec_esp	Length	HMAC Key	HMAC Data	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	HMAC In	HMAC Out	<i>undefined</i>
0001_1  ccmp	Length	nil	AES-Ctx	AES Key	In FIFO	In FIFO	Out FIFO	AES-Ctx-Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	HMAC In	HMAC Out	<i>undefined</i>

Table 12-9. Descriptor Format by Type (continued)

Descriptor Type	field type	Pointer Dword1	Pointer Dword2	Pointer Dword 3	Pointer Dword4	Pointer Dword 5	Pointer Dword 6	Pointer Dword 7
0010_1 srtp	Length	HMAC Key	AES-Ctx	AES Key	In FIFO	Out FIFO	HMAC Out	AES-Ctx-Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	In FIFO	In FIFO	nil	<i>undefined</i>
0011_1 pkbuild	Length	A0	A1	A2	A3	B0	B1	'Build'
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0100_1 pkptmul	Length	N	E	'Build'	B1 Out	B2 Out	B3 Out	nil
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
0101_1 pkptadd	Length	N	'Build'	B2	B3	B1 Out	B2 Out	B3 Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	nil	nil	nil	<i>undefined</i>
1000_1 outbound tls_ssl_block	Length	MAC Key	Cipher IV	Cipher Key	In FIFO Auth & Cipher	In FIFO Cipher Only	Out FIFO	Cipher IV Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	In FIFO Auth only	MAC Out	nil	<i>undefined</i>
1000_1 inbound tls_ssl_block	Length	MAC Key	Cipher IV	Cipher Key	nil	In FIFO Auth & Cipher	Out FIFO	Cipher IV Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	In FIFO Auth only	MAC In	MAC Out	<i>undefined</i>
1001_1 outbound tls_ssl_stream	Length	MAC Key	Cipher IV	Cipher Key	In FIFO Auth & Cipher	In FIFO Cipher Only	Out FIFO	Cipher IV Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	In FIFO Auth only	MAC Out	nil	<i>undefined</i>
1001_1 inbound tls_ssl_stream	Length	MAC Key	Cipher IV	Cipher Key	nil	In FIFO Auth & Cipher	Out FIFO	Cipher IV Out
	Extent	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	In FIFO Auth only	MAC In	MAC Out	<i>undefined</i>
1010_1 raid_xor	Length	nil	nil	nil	In 1 (opt)	In 2	In 3	Out
	Extent	nil	nil	nil	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>
others		Reserved						

## 12.4 Execution Units

Execution unit (EU) is the term used for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE 802.11i processing, and can work together to perform high level cryptographic tasks.

The following execution units are used in the SEC:

- Public key execution unit (PKEU)
- Data encryption standard execution unit (DEU)
- Advanced encryption standard execution unit (AESU) implementing the Rijndael symmetric key cipher.
- ARC four execution unit (AFEU)
- Message digest execution unit (MDEU)
- Kasumi (F8/F9) execution unit (KEU)
- One private on-chip random number generator (RNG)

Working together, the EUs can perform high-level cryptographic tasks, such as IPsec encapsulating security protocol (ESP) and digital signature. The remainder of this section provides details about the execution units themselves.

### 12.4.1 Public Key Execution Unit (PKEU)

This section contains details about the public key execution unit (PKEU), including modes of operation, status and control registers, and parameter RAMs.

Most of the registers described here are not normally accessed by the host. They are documented here mainly for debug purposes. In typical operation, the PKEU is used through channel-controlled access, which means that most reads and writes of PKEU registers are directed by the SEC channels. Driver software might perform host-controlled register accesses on only a few registers for initial configuration and error handling.

#### 12.4.1.1 PKEU Mode Register (PKEUMR)

This register specifies the internal PKEU routine to be executed. The mode register is cleared when the PKEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

Figure 12-8 shows the PKEU mode register, and Table 12-10 lists the possible values for the ROUTINE field.

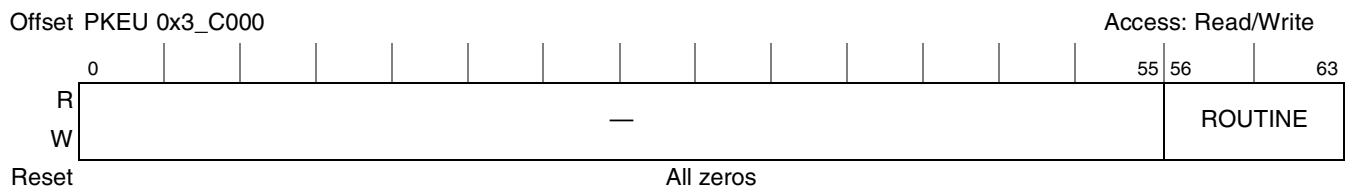


Figure 12-8. PKEU Mode Register

Table 12-10. PKEU[ROUTINE] Field Values

Mode [56-63]	Routine Name	Routine Description
0x00	RESERVED	Reserved
0x01	CLEARMEMORY	Clear memory
0x02	MOD_EXP	FP: Exponentiate mod N and deconvert from Montgomery format
0x03	MOD_R2MODN	FP: Compute Montgomery converter ( $R^2 \bmod N$ )
0x04	MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem ( $R_n R_p \bmod N$ )
0x05	EC_FP_AFF_PTMULT	FP EC: Multiply scalar times point in affine coordinates
0x06	EC_F2M_AFF_PTMULT	F2m EC: Multiply scalar times point in affine coordinates
0x07	EC_FP_PROJ_PTMULT	FP EC: Multiply scalar times point in projective coordinates
0x08	EC_F2M_PROJ_PTMULT	F2m EC: Multiply scalar times point in projective coordinates
0x09	EC_FP_ADD	FP EC: Add two points in projective coordinates
0x0A	EC_FP_DOUBLE	FP EC: Double a point in projective coordinates
0x0B	EC_F2M_ADD	F2m EC: Add two points in projective coordinates
0x0C	EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates
0x0D	F2M_R2	F2m: Compute Montgomery converter ( $R^2 \bmod N$ )
0x0E	F2M_INV	F2m: Invert mod N
0x0F	MOD_INV	FP: Invert mod N
0x10	MOD_ADD	FP: Add mod N
0x20	MOD_SUB	FP: Subtract mod N
0x30	MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format
0x40	MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format
0x50	F2M_ADD	F2m: Add mod N
0x60	F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format
0x70	F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format
0x80	RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, POLY_F2M_MULT1_MONT, and MOD_EXP)
0xFF	SPK_BUILD	Build PK data structure (data structure used by all elliptic curve routines)

### 12.4.1.2 PKEU Key Size Register (PKEUKSR)

The key size register reflects the number of significant bytes to be used from PKEU parameter memory E in performing modular exponentiation or elliptic curve point multiplication. The range of values for this register, when performing either modular exponentiation or elliptic curve point multiplication, is from 1 to 256. Specifying a key size outside of this range will cause a key size error (KSE) in the PKEU interrupt status register.

### 12.4.1.3 PKEU AB Size Register (PKEUABS)

The AB size register (Figure 12-9) represents the operand size for the specific operands whenever it is required. The unit of the value written into the AB size is in bits, even though internally the PKEU imposes a 32-bit alignment. Any data beyond the number of bits in the AB size register, either in A and B-RAM (operands) will be ignored. No error checking is performed whether the operand sizes are greater than the prime modulus or the field size and this may result in a wrong result. In other words, it is assumed that operands are modulo reduced before being written into the PKEU. Hence, the AB size must be less than or equal to data size for a correct result. If the AB size register is modified during processing, an error is generated.

An illegal data size error is generated as follows:

- For all non ECC routines a data size > 256 generates an illegal data size error.
- For all ECC routines a data size > 64 generates an illegal data size error.

An AB size = 0 (either intentionally written, or by ignoring and not writing at all) generates an illegal size error, except for routines that do not require an A or B operand such as the CLEAR\_MEM routine.

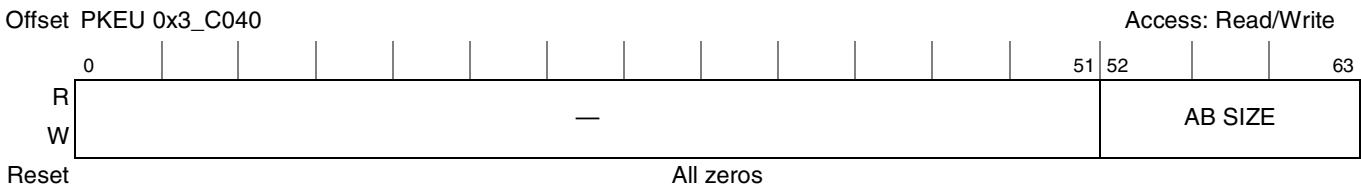


Figure 12-9. PKEU AB Size Register

### 12.4.1.4 PKEU Data Size Register (PKEUDSR)

The PKEU data size register, Figure 12-10, specifies, in bits, the size of the significant portion of the modulus or irreducible polynomial. Any value written to this register that is a multiple of 32 bits (for example, 128 bits, 160 bits,...), is represented internally as the same value (128 bits, 160 bits,...). Any value written that is not a multiple of 32 bits (for example, 132 bits, 161 bits,...), is represented internally as the next larger 32 bit multiple (160 bits, 196 bits,...). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 97 bits (internally 128 bits). The maximum size to operate properly is 2048 bits. A value in bits larger than 2048 results in a data size error.

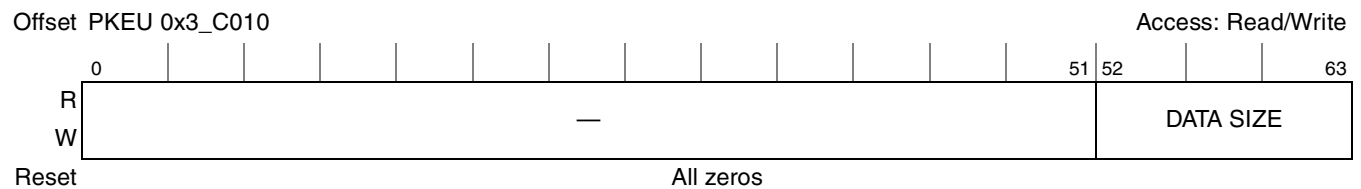


Figure 12-10. PKEU Data Size Register







Table 12-13 describes PKEU interrupt status register fields.

**Table 12-13. PKEU interrupt Status Register Field Descriptions**

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error. Indicates that the inversion routine has a zero operand. 0 No inversion error detected 1 Inversion error detected
51	IE	Internal error. An internal processing error was detected while the PKEU was operating. 0 No error detected 1 Internal error <b>Note:</b> This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the interrupt control register or by resetting the PKEU.
52	—	Reserved
53	CE	Context error. A PKEU key register, the key size register, the data size register, or mode register was modified while the PKEU was operating. 0 No error detected 1 Context error
54	KSE	Key size error. Value outside the bounds of 1–256 bytes was written to the PKEU key size register 0 No error detected 1 Key size error detected
55	DSE	Data size error. Value outside the bounds 97– 2048 bits was written to the PKEU data size register 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in the mode register. 0 No error detected 1 Mode error <b>Note:</b> Writing to reserved bits in a mode register is a likely source of error.
57	AE	Address error. Illegal read or write address was detected within the PKEU address space. 0 No error detected 1 Address error
58–63	—	Reserved

### 12.4.1.8 PKEU Interrupt Control Register (PKEUICR)

The PKEU interrupt control register controls the result of detected errors. For a given error (as defined in [Section 12.4.1.7, “PKEU Interrupt Status Register \(PKEUISR\)”](#)), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the PKEU interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Table 12-13 describes PKEU interrupt control register fields.

**Table 12-14. PKEU Interrupt Control Register Field Descriptions**

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error 0 Inversion error enabled 1 Inversion error disabled
51	IE	Internal error 0 Internal error enabled 1 Internal error disabled
52	—	Reserved
53	CE	Context error 0 Context error enabled 1 Context error disabled
54	KSE	Key size error 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error 0 Mode error enabled 1 Mode error disabled
57	AE	Address error 0 Address error enabled 1 Address error disabled
58–63	—	Reserved

### 12.4.1.9 PKEU EU Go Register (PKEUEUG)

The EU go register in the PKEU is used to indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the ROUTINE field, per the contents of the parameter memories listed below. This register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated.

### 12.4.1.10 PKEU Parameter Memories

The PKEU uses four 2048-bit memories to receive and store operands for the arithmetic operations the PKEU will be asked to perform. In addition, results are stored in one particular parameter memory.

Data addressing within these memories is big endian; that is, the most significant byte is stored in the lowest address.

#### 12.4.1.10.1 PKEU Parameter Memory A

This 2048 bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 512 bit memories, and it is used to specify particular curve parameters and input values.

#### 12.4.1.10.2 PKEU Parameter Memory B

This 2048 bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 512 bit memories, and it is used to specify particular curve parameters and input values, as well as to store result values.

#### 12.4.1.10.3 PKEU Parameter Memory E

This 2048 bit memory is non-segmentable, and stores the exponent for modular exponentiation, or the multiplier  $k$  for elliptic curve point multiplication. This memory space is write only; a read of this memory space causes an address error to be reflected in the PKEU interrupt status register.

#### 12.4.1.10.4 PKEU Parameter Memory N

This 2048 bit memory is non-segmentable, and stores the modulus for modular arithmetic and  $F_p$  elliptic curve routines. For  $F_{2^m}$  elliptic curve routines, this memory stores the irreducible polynomial.

### 12.4.2 Data Encryption Standard Execution Unit (DEU)

This section contains details about the data encryption standard execution unit (DEU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here are not normally accessed by the host. They are documented here mainly for debug purposes. In typical operation, the DEU is used through channel-controlled access, which means that most reads and writes of DEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.

#### 12.4.2.1 DEU Mode Register (DEUMR)

The DEUMR mode register contains 3 bits which are used to program DEU operation.

The mode register is cleared when the DEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

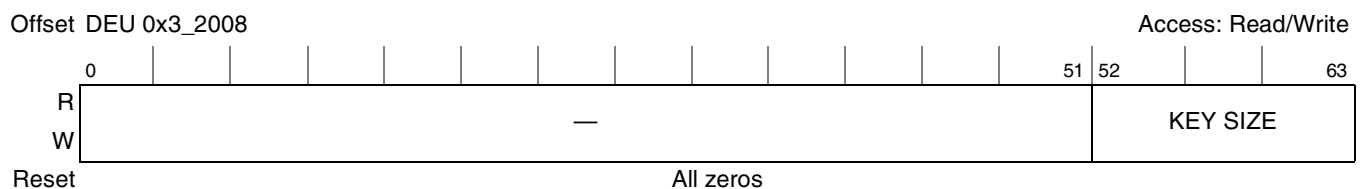
Table 12-15 describes the DEU mode register fields.

**Table 12-15. DEU Mode Register Field Descriptions**

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–54	—	Reserved
55	—	Reserved
The following bits are controlled through the MODE0 field of the descriptor header.		
56–60	—	Reserved
61	CE	CBC/ECB. If set, DEU operates in cipher-block-chaining mode. If not set, DEU operates in electronic codebook mode. 0 ECB mode 1 CBC mode
62	TS	Triple/single DES. If set, DEU operates the triple DES algorithm; if not set, DEU operates the single DES algorithm. 0 Single DES 1 Triple DES
63	ED	Encrypt/decrypt. If set, DEU operates the encryption algorithm; if not set, DEU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption

### 12.4.2.2 DEU Key Size Register (DEUKSR)

This value indicates the number of bytes of key memory that should be used in encrypting or decrypting. If the DEU mode register is set for single DES, any value other than 8 bytes will automatically generate a key size error in the DEU interrupt status register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1=K3) or 24 bytes (168 bits for 3-key triple DES) will generate an error. Triple DES always uses K1 to encrypt, Key2 to decrypt, K3 to encrypt.



**Figure 12-14. DEU Key Size Register**

Table 12-16 shows the legal values for DEU key size.

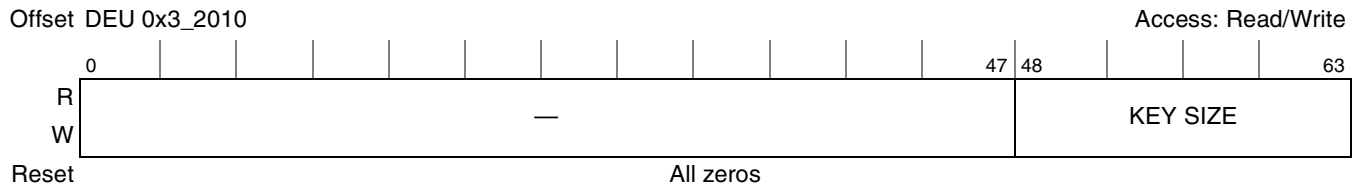
**Table 12-16. DEU Key Size Register Field Descriptions**

Bits	Name	Description
0–51	—	Reserved
52–63	Key Size	8 bytes = 0x08 (only legal value if mode is single DES.) 16 bytes= 0x10 (for 2 key 3DES, K1 = K3) 24 bytes= 0x18 (for 3 key 3DES)

### 12.4.2.3 DEU Data Size Register (DEUDSR)

This register, shown in [Figure 12-15](#), stores the number of bits in the final message block, which must be 64. All data to be processed by the DEU must be a multiple of the DES algorithm block size of 64 bits; the DEU does not automatically pad messages out to 64-bit blocks. If a data size that is not a multiple of 64 bits is written, a data size error will be generated. Only bits 58–63 are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

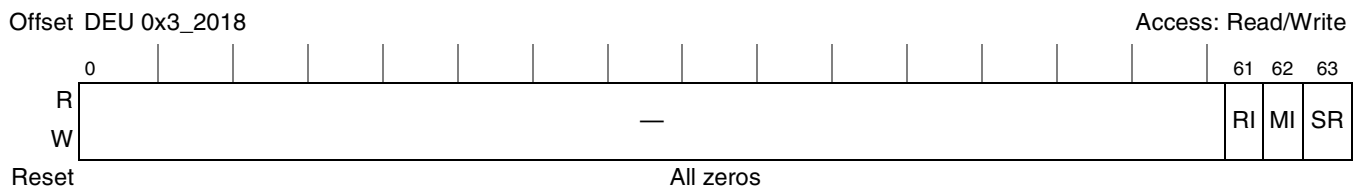
This register is cleared when the DEU is reset or re-initialized.



**Figure 12-15. DEU Data Size Register**

### 12.4.2.4 DEU Reset Control Register (DEURCR)

This register, shown in [Figure 12-16](#), allows three levels reset of just DEU, as defined by the three self-clearing bits:



**Figure 12-16. DEU Reset Control Register**

[Table 12-17](#) describes DEU reset control register fields.

**Table 12-17. DEU Reset Control Register Field Descriptions**

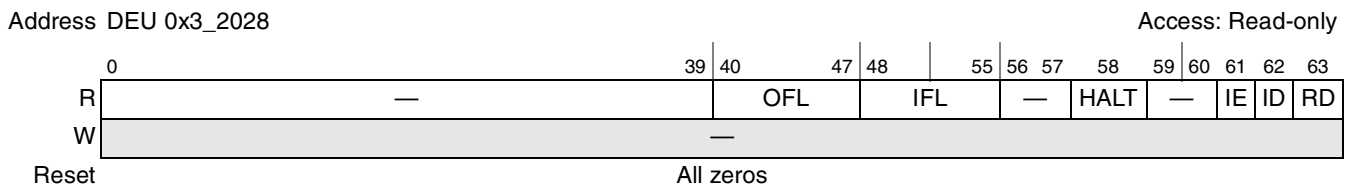
Bits	Names	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes DEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the DEU interrupt status register. 0 Don't reset 1 Reset interrupt logic

**Table 12-17. DEU Reset Control Register Field Descriptions (continued)**

Bits	Names	Description
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the DEU status register 0 Don't reset 1 Reset most of DEU
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU enters a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the DEU status register indicates when this initialization routine is complete 0 Don't reset 1 Full DEU reset

### 12.4.2.5 DEU Status Register (DEUSR)

This status register, shown in [Figure 12-17](#), contains 6 fields which reflect the state of DEU internal signals. The DEU status register is read-only. Writing to this location results in an address error being reflected in the DEU interrupt status register.



**Figure 12-17. DEU Status Register**

[Table 12-18](#) describes the DEU status register bit settings.

**Table 12-18. DEU Status Register Field Descriptions**

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the DEU has halted due to an error. 0 DEU not halted 1 DEU halted <b>Note:</b> Because the error causing the DEU to stop operating may be masked before reaching the interrupt status register, the DEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register ( <a href="#">Section 12.6.5.3, “Interrupt Status Register (ISR)”</a> ). 0 DEU is not signaling error 1 DEU is signaling error



Table 12-19. DEU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error <b>Note:</b> This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the interrupt control register or by resetting the DEU.
52	ERE	Early read error. The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. A DEU key register, the key size register, data size register, mode register, or IV register was modified while DEU was performing encryption. 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU key size register 0 No error detected 1 Key size error
55	DSE	Data size error (DSE). A value was written to the DEU data size register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	IFU	Input FIFO underflow. The DEU input FIFO was read while empty. 0 No error detected 1 Input FIFO has had underflow error
61	IFO	Input FIFO overflow. The DEU input FIFO was pushed while full. 0 No error detected 1 Input FIFO has overflowed <b>Note:</b> When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.

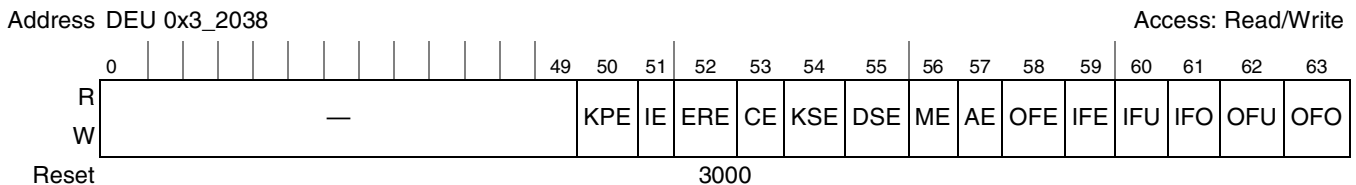


**Table 12-19. DEU Interrupt Status Register Field Descriptions (continued)**

Bits	Name	Description
62	OFU	Output FIFO underflow. The DEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	OFO	Output FIFO overflow. The DEU output FIFO was pushed while full. 0 No error detected 1 Output FIFO has overflowed

### 12.4.2.7 DEU Interrupt Control Register (DEUICR)

The interrupt control register controls the result of detected errors. For a given error (as defined in [Section 12.4.2.6, “DEU Interrupt Status Register \(DEUISR\)”](#)), if the corresponding bit in this register is set, then the error is ignored; no bit is set in the DEU interrupt status register, and no error interrupt occurs. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

**Figure 12-19. DEU Interrupt Control Register****Table 12-20. DEU Interrupt Control Register Field Descriptions**

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity error enabled 1 Key parity error disabled
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The DEU IV register was read while the DEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. A DEU key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled

Table 12-20. DEU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for Triple DES) was written to the DEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error (DSE): A value that is not a multiple of 64 bits was written to the DEU data size register. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	IFU	Input FIFO underflow. The DEU input FIFO was read while empty. 0 Input FIFO Underflow error enabled 1 Input FIFO Underflow error disabled
61	IFO	Input FIFO overflow. The DEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled <b>Note:</b> When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	OFO	Output FIFO overflow. The DEU output FIFO was pushed while full. 0 Output FIFO overflow error enabled 1 Output FIFO overflow error disabled

### 12.4.2.8 DEU EU Go Register (DEUEUG)

The EU go register in the DEU is used to indicate a DES operation may be completed. After the final message block is written to the input FIFO, the EU go register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 64) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated. Writing to

this register is merely a trigger causing the DEU to process the final block of a message, allowing it to signal DONE.

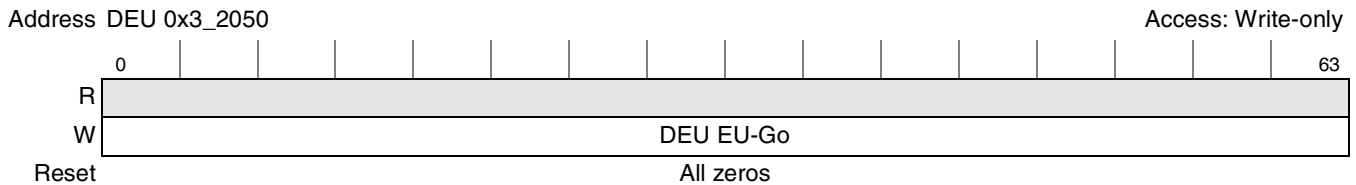


Figure 12-20. DEU EU Go Register

### 12.4.2.9 DEU IV Register (DEUIV)

For CBC mode, the initialization vector is written to and read from the DEU IV register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

### 12.4.2.10 DEU Key Registers 1–3 (DEUK $n$ )

The DEU uses three write-only key registers to perform encryption and decryption. In single DES mode, only key register 1 may be written. The value written to key register 1 is simultaneously written to key register 3, auto-enabling the DEU for 112-bit triple DES if the key size register indicates 2 key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit triple DES, key register 1 must be written first, followed by the write of key register 2, and then key register 3.

Reading any of these memory locations generates an address error interrupt.

### 12.4.2.11 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the DEU FIFO address space causes the 64-bit-word to be pushed onto the DEU input FIFO, and a read from anywhere in the DEU FIFO Address space causes a 64-bit-word to be popped off of the DEU output FIFO. Overflows and underflows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

## 12.4.3 ARC Four Execution Unit (AFEU)

This section contains details about the ARC four execution unit (AFEU), including modes of operation, status and control registers, S-box memory, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the AFEU is used through channel-controlled access, which means that most reads and writes of AFEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.

### 12.4.3.1 AFEU Mode Register (AFEUMR)

Shown in [Figure 12-21](#), the AFEU mode register contains three bits which are used to program the AFEU. The mode register is cleared when the AFEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

### 12.4.3.2 Host-Provided Context via Prevent Permute

In the default mode of operation, the host provides the key and key size to the AFEU. The initial memory values in the S-box are permuted with the key to create new S-box values, which are used to encrypt the plaintext.

If the prevent permute mode bit is set, the AFEU does not require a key. Rather, the host writes the context to the AFEU and message processing occurs using the provided context. This mode is used to resume processing of a message using the already permuted S-box. The context may be written through the FIFO if the context source mode bit is set. The AFEU context is 259 bytes long, and must be in the format provided by the dump context function. (See [Section 12.4.3.2.1, “Dump Context.”](#))

#### 12.4.3.2.1 Dump Context

This mode may be independently specified in addition to host-provided context mode. In this mode, once message processing is complete and the output data is read, the AFEU will make the current context data available for reads via the output FIFO. The AFEU context is 259 bytes long.

#### NOTE

After the initial key permute to generate a context for an AFEU encrypted session, all subsequent messages will re-use that context, such that it is loaded, modified during the encryption, and unloaded, similar to the use of a CBC initialization vector in DES operations. A new context is generated (via key permute) according to a re-keying interval specified by the security protocol. Context should never be loaded to encrypt a message if a key is loaded and permuted at the same time.



Figure 12-21. AFEU Mode Register

Table 12-21 describes AFEU mode register fields.

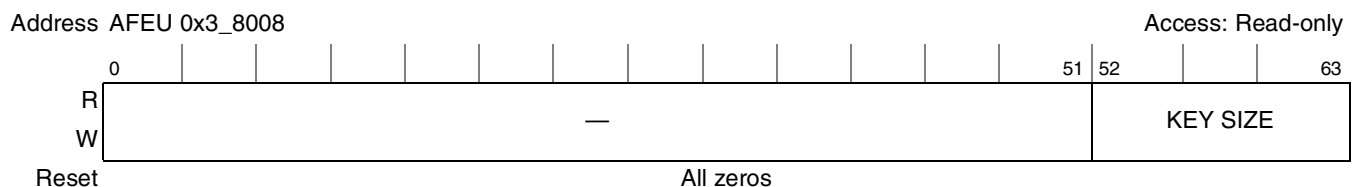
**Table 12-21. AFEU Mode Register Field Descriptions**

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–55	—	Reserved
The following bits are controlled through the MODE0 field of the descriptor header.		
56–60	—	Reserved
61	CS	Context source. If set, this causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the context registers. Context source is only checked if the prevent permute bit is set. 0 Context not from FIFO 1 Context from input FIFO
62	DC	Dump context. If set, this causes the context to be moved from the S-box to the output FIFO following assertion AFEU's done interrupt. 0 Do not dump context 1 After cipher, dump context
63	PP	Prevent permute. Normally, the AFEU receives a key and uses that information to randomize the S-box. If re-using a context from a previous descriptor, this bit should be set to prevent AFEU from re-performing this permutation step. 0 Perform S-box permutation 1 Do not permute

### 12.4.3.3 AFEU Key Size Register (AFEUKSR)

This value indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in the key size register is ignored. This register is cleared when the AFEU is reset or re-initialized. If the key size specified is less than 1 or greater than 16, a key size error is generated. If the key size register is modified during processing, a context error is generated. Note: Although the AFEU supports key lengths as short as 1 byte, a 1 byte key offers little security. Most uses of ARC-4 specify keys of 5–16 bytes.

Figure 12-22 depicts the AFEU key size register.



**Figure 12-22. AFEU Key Size Register**

**NOTE**

The device driver will create properly formatted descriptors for situations requiring a key permute prior to ciphering. When operating the SEC as a slave (typically for debug), the user must set the AFEU mode register to perform a permute with key function, then write the key data to AFEU key registers, then write the key size to the key size register. The AFEU starts permuting the memory with the contents of the key registers immediately after the key size is written.

**12.4.3.4 AFEU Context/Data Size Register (AFEUDSR)**

The AFEU context/data size register, shown in Figure 12-23, stores the number of bits in the final message block. This value controls how much data will be processed from the last block. The last message block must be a multiple of 8, from 8 to 64. If a data size that is not a multiple of 8 bits is written, a data size error is generated. A data size of 0 is illegal and results in the associated crypto-channel locking, requiring a crypto-channel and AFEU reset. Only bits 61–63 are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

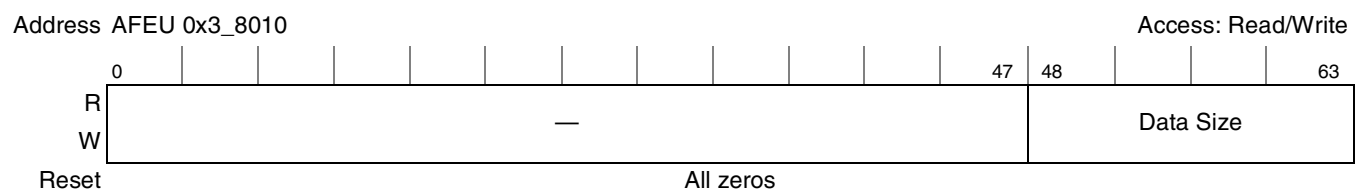
The context/data size register is also used to specify the context size, when context is used. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

**NOTE**

In slave mode, when reloading an existing context, the user must write the context to the input FIFO, then write the context size (always 2072 bits). The write of the context size indicates to the AFEU that all context has been loaded. The user then writes the message data size to the context/data size register. After this write, the user may begin writing message data to the FIFO.

Writing to this register signals the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

This register is cleared when the AFEU is reset or re-initialized.

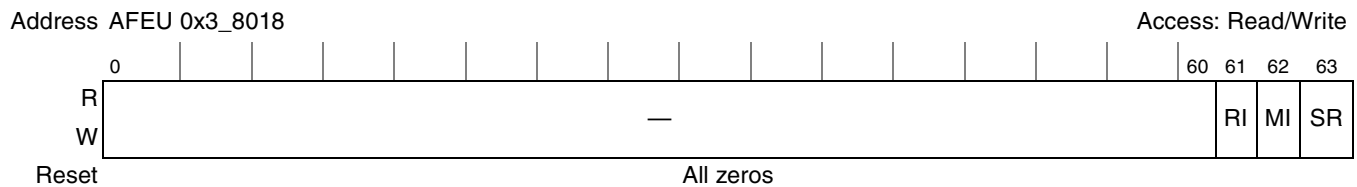


**Figure 12-23. AFEU Context/Data Size Register**

**12.4.3.5 AFEU Reset Control Register (AFEURCR)**

This register, as shown in Figure 12-24, allows 3 levels of reset that effect the AFEU only, as defined by 3 self-clearing bits. Note that the AFEU executes an internal reset sequence for hardware reset,

SW\_RESET, or module initialization, which performs proper initialization of the S-box. To determine when this is complete, poll the RESET\_DONE bit in the AFEU status register.



**Figure 12-24. AFEU Reset Control Register**

Table 12-22 describes AFEU reset control register fields.

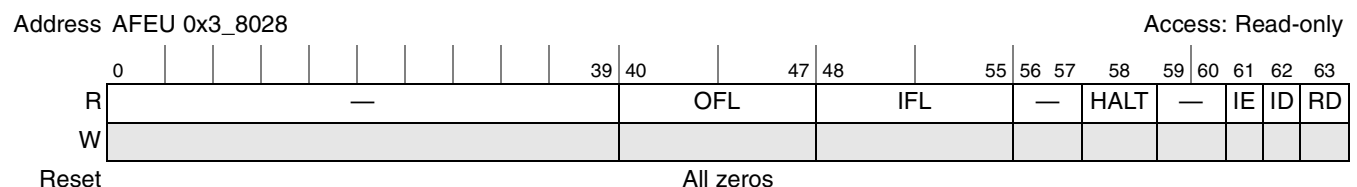
**Table 12-22. AFEU Reset Control Register Field Descriptions**

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes AFEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the AFEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 Do not reset 1 Reset most of AFEU
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for AFEU. All registers and internal states are returned to their defined reset state. On negation of SW_RESET, the AFEU enters a routine to perform proper initialization of the S-box. 0 Do not reset 1 Full AFEU reset

### 12.4.3.6 AFEU Status Register (AFEUSR)

This status register, shown in Figure 12-25, contains 6 bits that reflect the state of the AFEU internal signals.

The AFEU status register is read-only. Writing to this location results in an address error being reflected in the AFEU interrupt status register.



**Figure 12-25. AFEU Status Register**

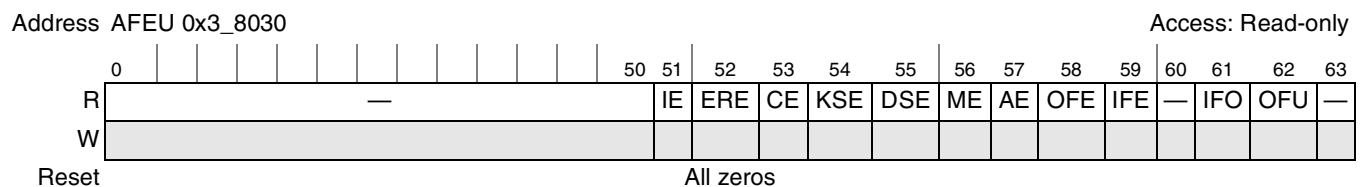
Table 12-23 describes the AFEU status register fields.

**Table 12-23. AFEU Status Register Field Descriptions**

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the AFEU has halted due to an error. 0 AFEU not halted 1 AFEU halted <b>Note:</b> Because the error causing the AFEU to stop operating may be masked before reaching the interrupt status register, the AFEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR”). 0 AFEU is not signaling error 1 AFEU is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR”). 0 AFEU is not signaling done 1 AFEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that AFEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done <b>Note:</b> Reset done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

### 12.4.3.7 AFEU Interrupt Status Register (AFEUISR)

The interrupt status register, shown in Figure 12-26, tracks the state of possible errors, if those errors are not masked, via the AFEU interrupt control register.



**Figure 12-26. AFEU Interrupt Status Register**



Table 12-24 describes AFEU interrupt status register fields.

**Table 12-24. AFEU Interrupt Status Register Field Descriptions**

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error
52	ERE	Early read error. The AFEU context memory or control was read while the AFEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. The AFEU mode register, key register, key size register, data size register, or context memory is modified while AFEU processes data. 0 No error detected 1 Context error
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 No error detected 1 Key size error
55	DSE	Data size error. A value that is not a multiple of 8 bits was written to the AFEU data size register: 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The AFEU input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU input FIFO was pushed while full. 1 Input FIFO has overflowed 0 No error detected <b>Note:</b> When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the AFEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The AFEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

### 12.4.3.8 AFEU Interrupt Control Register (AFEUICR)

The interrupt control register, shown in Figure 12-27, controls the result of detected errors. For a given error (as defined in Section 12.4.3.7, “AFEU Interrupt Status Register (AFEUISR)”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

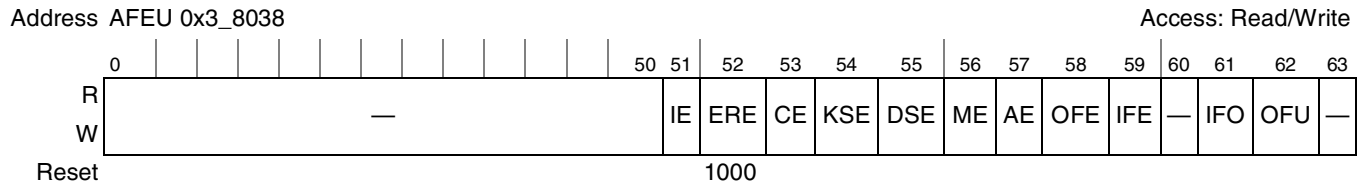


Figure 12-27. AFEU Interrupt Control Register

Table 12-25 describes AFEU interrupt control register fields.

Table 12-25. AFEU Interrupt Control Register Field Descriptions

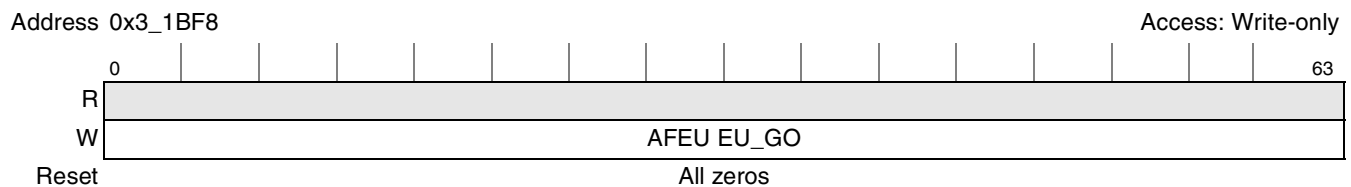
Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The AFEU register was read while the AFEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. An AFEU key register, the key size register, data size register, mode register, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value was written to the AFEU data size register: 0 Data Size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled

**Table 12-25. AFEU Interrupt Control Register Field Descriptions (continued)**

Bits	Name	Description
59	IFE	Input FIFO error. The AFEU input FIFO was detected non-empty upon generation of done interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The AFEU output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

### 12.4.3.9 AFEU EU Go Register (AFEUEUG)

The EU go register in the AFEU, displayed in [Figure 12-28](#), is used to signal the AFEU that all data to be processed has been written to the input FIFO. This allows the AFEU to perform special processing when it reaches the last block of data. Before this register is written, the AFEU will not process the last block of data in its input FIFO. After this register is written, the AFEU continues to perform normal processing on all but the last block of data, then it goes on to process the last block, using the value in the data size register to determine how much of the block to process. The data size register specifies the number of bits to process, which is a multiple of 8, from 8 to 64. After processing of the last block is completed, the AFEU signals DONE. If the dump context bit in the AFEU mode register is set, the context is written to the output FIFO following the last message word. A read of the AFEUEUG register always returns a zero value.

**Figure 12-28. AFEU EU Go Register**

### 12.4.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

#### 12.4.3.10.1 AFEU Context Memory

The S-box memory consists of 32 64-bit words, each readable and writable. The S-box contents should not be written with data unless it was previously read from the S-box. Context data may only be written if the prevent permutation mode bit is set (see [Figure 12-21](#)) and the context data must be written prior to the message data. If the context registers are written during message processing or the prevent permutation bit is not set, a context error will be generated. Reading this memory while the module is not done generates an error interrupt.

### 12.4.3.10.2 AFEU Context Memory Pointer Register

The context memory pointer register holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC-4 algorithm. If this register is written during message processing, a context error will be generated.

When performing ARC-4 operations, the user has the option of performing a new S-box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing of the next packet. The S-box contents (256 bytes) plus the three bytes of the context memory pointers are unloaded and reloaded via the AFEU FIFOs.

AFEU context consists of the contents of the S-box, as well as three counter values, which indicate the next values to be used from the S-box. Context must be loaded in the same order in which it was unloaded.

### 12.4.3.11 AFEU Key Registers 0–1 (AFEUK $n$ )

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-box, in conjunction with the AFEU key size register. AFEU performs permutation starting with the first byte of key register 1, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

### 12.4.3.12 AFEU FIFOs

The AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the AFEU FIFO address space causes the 64-bit-word to be pushed onto the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space causes a 64-bit-word to be popped off of the AFEU output FIFO. Overflows and underflows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

## 12.4.4 Message Digest Execution Unit (MDEU)

This section contains details about the message digest execution unit (MDEU), including modes of operation, status and control registers, and FIFOs.

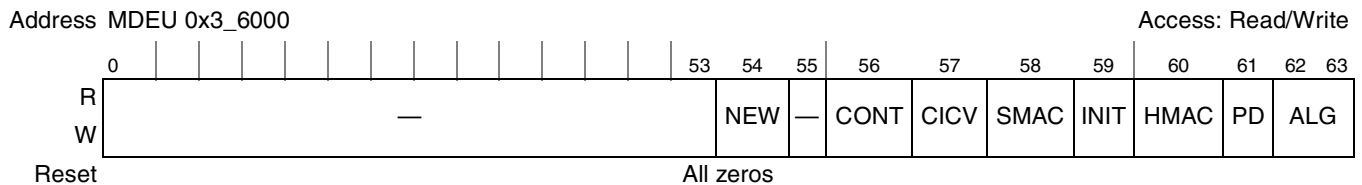
Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the MDEU is used through channel-controlled access, which means that most reads and writes of MDEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.

### 12.4.4.1 MDEU Mode Register (MDEUMR)

The MDEU mode register is used to program the function of the MDEU. Bits 56–63 of this register are specified by the user through the MODE0 or MODE1 field of the descriptor header. The remaining bits are supplied by the channel and thus are not under direct user control.

The MDEU mode register has two configurations, determined by the value of the NEW bit (see [Figure 12-29](#) and [Figure 12-30](#)). The new configuration (NEW = 1) is used only by TLS/SSL descriptor types (1000\_1, 1001\_). The old configuration (NEW = 0) is used by all other descriptor types and is backward compatible with SEC 2.0.

The mode register is cleared when the MDEU is reset or re-initialized. Setting a reserved mode bit generated a data error. If the mode register is modified during processing, a context error is generated.



**Figure 12-29. MDEU Mode Register in Old Configuration (NEW = 0)**

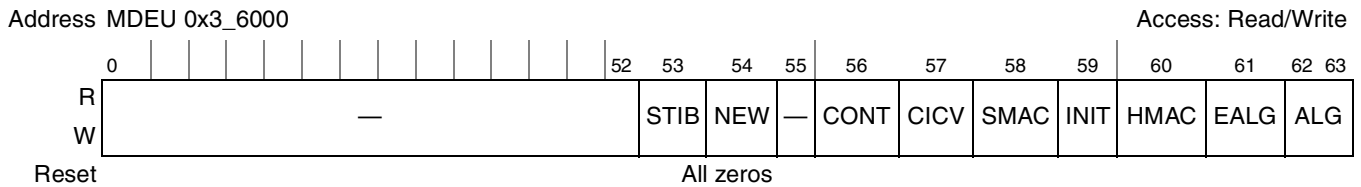
[Table 12-26](#) describes the MDEU mode register fields in the old configuration.

**Table 12-26. MDEU Mode Register in Old Configuration (NEW = 0)**

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–53	—	Reserved
54	NEW=0	Determines the configuration of the MDEU mode register. This table shows the configuration for NEW=0.
55	—	Reserved. Must be set to zero.
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue. Most operations require this bit to be cleared. It is set only when the data to be hashed is spread across multiple descriptors. The value programmed in PD must be opposite to the value in this bit. 0 Perform autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare integrity check values. 0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register. Only applicable to descriptor types that provide for reading an ICV in value.
58	SMAC	Specifies whether to perform an SSL-MAC operation. 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit must be 0 on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.

**Table 12-26. MDEU Mode Register in Old Configuration (NEW = 0) (continued)**

Bits	Name	Description
60	HMAC	Specifies whether to perform an HMAC operation: 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	PD	If set, configures the MDEU to automatically pad partial message blocks. This bit must be programmed opposite to the CONT bit. 0 Do not autopad. 1 Perform automatic message padding whenever an incomplete message block is detected.
62–63	ALG	Message digest algorithm selection 00 SHA-160 algorithm (full name for SHA-1) 01 SHA-256 algorithm 10 MD5 algorithm 11 SHA-224 algorithm



**Figure 12-30. MDEU Mode Register in New Configuration (NEW = 1)**

Table 12-27 describes MDEU mode register fields in the new configuration.

**Table 12-27. MDEU Mode Register in New Configuration (NEW = 1)**

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–52	—	Reserved
53	STIB	SSL/TLS inbound, block cipher. 0 Normal operation. 1 Special operation only for SSL/TLS inbound, block cipher. Upon receiving an EU_Go indication, the MDEU performs a calculation involving the last valid byte of data written into its input FIFO (which is pad length) to compute a final data size. The MDEU then processes the amount of data specified by this data size, and completes the message digest.
54	NEW=1	Determines the configuration of the MDEU mode register. This table shows the configuration for NEW=1.
55	—	Reserved. Must be set to zero.
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue. Most operations will require this bit to be cleared. Set only when the data to be hashed is spread across multiple descriptors. 0 Perform autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.

**Table 12-27. MDEU Mode Register in New Configuration (NEW = 1) (continued)**

Bits	Name	Description
57	CICV	Compare integrity check values. 0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register.
58	SMAC	Specifies whether to perform an SSL-MAC operation: 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit is set on all but the first descriptor. 1 Perform an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation: 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	EALG	The EALG (extended algorithm bit) and ALG (algorithm) bits together specify the message digest algorithm, as follows: 000 SHA-160 algorithm (full name for SHA-1) 001 SHA-256 algorithm 010 MD5 algorithm 011 SHA-224 algorithm
62–63	ALG	

#### 12.4.4.2 Recommended Settings for MDEU Mode Register

The most common task likely to be executed via the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPsec, and TLS. The SSL 3.0 protocol uses a slightly different SSL-MAC. If an HMAC or SSL-MAC is to be performed using a single descriptor (with the MDEU acting as sole or secondary EU), the following mode register bit settings should be used:

**Table 12-28. Mode Register —HMAC or SSL-MAC Generated by Single Descriptor**

Bits	Field	Value	
		for HMAC	for SSL-MAC
56	CONT	0 (off)	0 (off)
58	SMAC	0(on)	1(on)
59	INIT	1(on)	1(on)
60	HMAC	1(on)	0(on)

To generate an HMAC for a message that is spread across a sequence of descriptors, the following mode register bit settings should be used:

**Table 12-29. Mode Register —HMAC Generated Across a Sequence of Descriptors**

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
56	CONT	1 (on)	1 (on)	0 (off)
59	INIT	1 (on)	0 (off)	0 (off)
60	HMAC	1 (on)	0 (off)	1 (on)

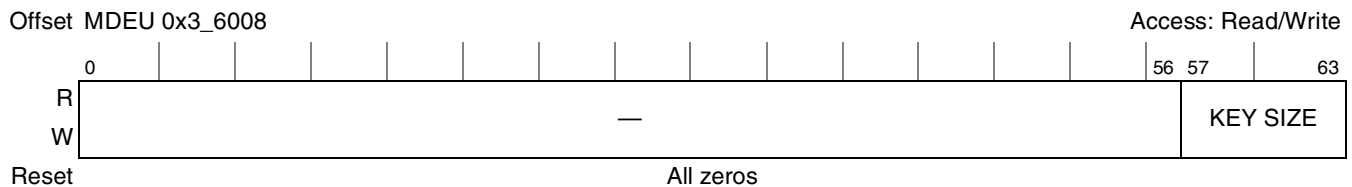
All descriptors other than the final descriptor must output the intermediate message digest for the next descriptor to reload as MDEU context.

SSL-MAC operations cannot be spread across a sequence of descriptors.

Additional information on descriptors can be found in [Section 12.3, “Descriptor Overview.”](#)

#### 12.4.4.3 MDEU Key Size Register (MDEUKSR)

Shown in [Figure 12-31](#), this value indicates the number of bytes of key memory that should be used in HMAC generation. MDEU supports at most 64 bytes of key. MDEU generates a key size error if the value written to this register exceeds 64 bytes.

**Figure 12-31. MDEU Key Size Register**

#### 12.4.4.4 MDEU Data Size Register (MDEUDSR)

The MDEU data size register, shown in [Figure 12-32](#), indicates the number of bits of data to be processed.

The data size field is a 21-bit signed number. Values written to this register are added to the current register value. Multiple writes are allowed. The MDEU processes data when there is a positive value in this register and there is data available in the MDEU input FIFO. (Negative values can arise in inbound processing, when it is necessary to hold back data from the MDEU until the pad length has been decrypted.)

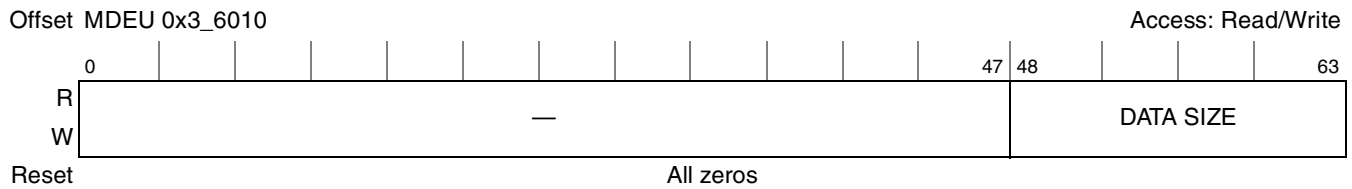
Since the MDEU does not support bit offsets, bits 61–63 must be written as 0, and are always read as zero. Furthermore, when the CONT bit of the MDEU mode register is high, the data size must be a multiple of the 512-bit block size (i.e. bits 55–63 must be written as 0). Violating either of these conditions causes a data size error (DSE in the MDEU interrupt status register).

This register is cleared when the MDEU is reset or re-initialized. At the end of processing, its contents have been decremented down to zero (unless there is an error interrupt).

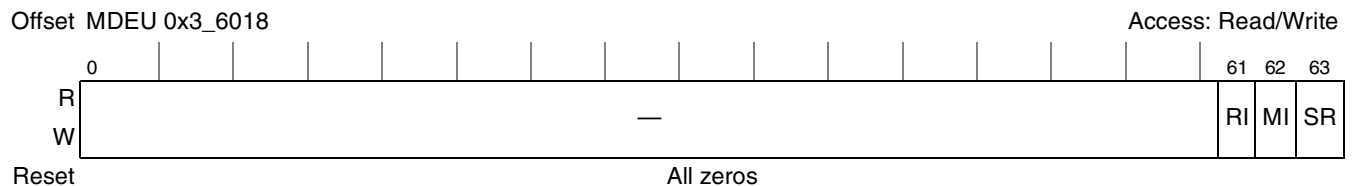


**NOTE**

Writing to the data size register will allow the MDEU to enter auto-start mode. Therefore, the required context registers must be written prior to writing the data size.

**Figure 12-32. MDEU Data Size Register****12.4.4.5 MDEU Reset Control Register (MDEURCR)**

This register, shown in [Figure 12-33](#), allows three levels of reset for the MDEU, as defined by the three self-clearing bits.

**Figure 12-33. MDEU Reset Control Register**

[Table 12-30](#) describes MDEU reset control register fields.

**Table 12-30. MDEU Reset Control Register Field Descriptions**

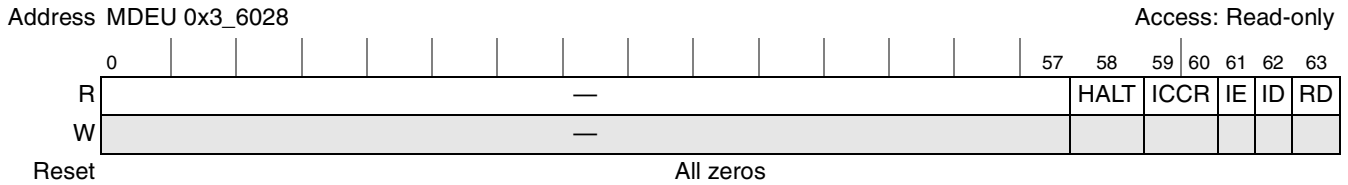
Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes MDEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the MDEU interrupt status register. 0 No reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the MDEU interrupt control register remains unchanged. 0 No reset 1 Reset most of MDEU
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset

**12.4.4.6 MDEU Status Register (MDEUSR)**

This status register, as seen in [Figure 12-34](#), reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding, key padding, etc., and are not important to the user; however the user should be aware that reads of this register,

especially during processing, are likely to return non-zero values for many bits between 0–57. The four signals shown are those which are most likely to be of interest to the user.

The MDEU status register is read-only. Writing to this location results in address error being reflected in the MDEU interrupt status register.



**Figure 12-34. MDEU Status Register**

Table 12-31 describes the MDEU status register fields.

**Table 12-31. MDEU Status Register Field Descriptions**

Bits	Name	Description
0–57	—	Reserved
58	HALT	Halt. Indicates that the MDEU has halted due to an error. 0 MDEU not halted 1 MDEU halted <b>Note:</b> Because the error causing the MDEU to stop operating may be masked before reaching the interrupt status register, the MDEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved <b>Note:</b> A passed or failed result is generated only if ICV checking is enabled.
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register ( <a href="#">Section 12.6.5.3, “Interrupt Status Register (ISR)”</a> ). 0 MDEU is not signaling error 1 MDEU is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register ( <a href="#">Section 12.6.5.3, “Interrupt Status Register (ISR)”</a> ). 0 MDEU is not signaling done 1 MDEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that MDEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done <b>Note:</b> Reset done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.





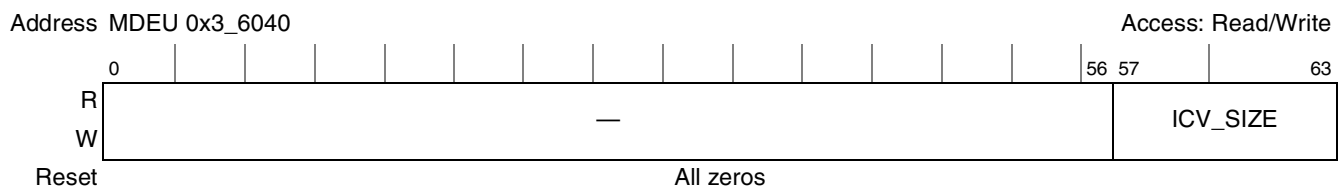
**Table 12-33. MDEU Interrupt Control Register Field Descriptions (continued)**

Bits	Name	Description
54	KSE	Key size error. A value outside the bounds 64 bytes was written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value was written to the MDEU data size register: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–60	—	Reserved
61	IFO	Input FIFO overflow. The MDEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62–63	—	Reserved

#### 12.4.4.9 MDEU ICV Size Register (MDEUICVSR)

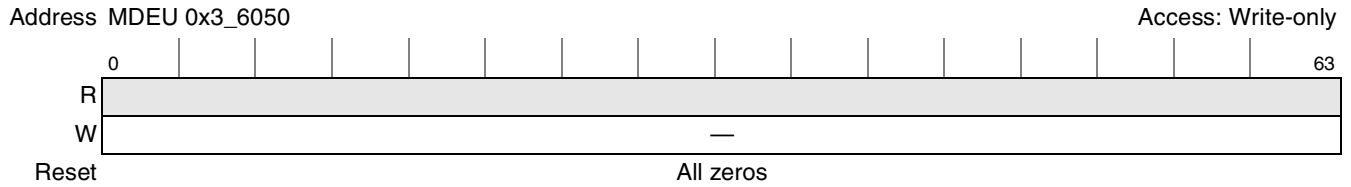
The MDEU ICV size register, shown in [Figure 12-37](#), stores the number of bytes of the ICV result to be compared if the MDEU performs ICV comparison. (See [Section 12.4.4.1](#), “MDEU Mode Register (MDEUMR).”)

This register is cleared when the MDEU is reset or re-initialized.

**Figure 12-37. MDEU ICV Size Register**

#### 12.4.4.10 MDEU EU Go Register (MDEUEUG)

The EU go register in the MDEU, see [Figure 12-38](#), is used to indicate an authentication operation may be completed. After the final message block is written to the input FIFO, the EU go register must be written. The value in the data size register is used to determine how many bits of the final message block (always 512) is processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.



**Figure 12-38. MDEU EU Go Register**

### 12.4.4.11 MDEU Context Registers

For the MDEU, the context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bit count.

#### **NOTE**

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done will generate an error interrupt.

After a power-on reset, all the MDEU context register values are cleared to 0. [Figure 12-39](#) shows how the MDEU context registers are initialized if the INIT bit is set in the MDEU mode register.

	0	31	32	63	
<b>Name</b>	<b>A</b>		<b>B</b>		Context offset 0x3_6100
MD-5	0x01234567		0x89ABCDEF		
SHA-1	0x67452301		0xEFCDAB89		
SHA-256	0x6A09E667		0xBB67AE85		
SHA-224	0xC1059ED8		0x367CD507		
<b>Name</b>	<b>C</b>		<b>D</b>		Context offset 0x3_6108
MD-5	0xFEDCBA98		0x76543210		
SHA-1	0x98BADCFE		0x10325476		
SHA-256	0x3C6EF372		0xA54FF53A		
SHA-224	0x3070DD17		0xF70E5939		
<b>Name</b>	<b>E</b>		<b>F</b>		Context offset 0x3_6110
MD-5	0xF0E1D2C3		0x8C68059B		
SHA-1	0xC3D2E1F0		0x9B05688C		
SHA-256	0x510E527F		0x9B05688C		
SHA-224	0xFFC00B31		0x68581511		
<b>Name</b>	<b>G</b>		<b>H</b>		Context offset 0x3_6118
MD-5	0xABD9831F		0x19CDE05B		
SHA-1	0x1F83D9AB		0x5BE0CD19		
SHA-256	0x1F83D9AB		0x5BE0CD19		
SHA-224	0x64F98FA7		0xBEFA4FA4		
<b>Name</b>	<b>Message Length Count</b>				Context offset 0x3_6120
Reset	0				

Figure 12-39. MDEU Context Register

All registers are initialized, regardless of mode selected, however only the appropriate context register values are used in hash generation per the mode selected. The user typically doesn't care about the MDEU context register initialization values; however they are documented for completeness in the event the user reads these registers using host-controlled access. MDEU reset via the MDEU reset control register (Figure 12-33) or SEC global software reset (Figure 12-85) does not clear these registers.

#### 12.4.4.12 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

#### NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

### 12.4.4.13 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed. The input FIFO is multiple-addressable, but those multiple addresses point only to the write (push) end of the FIFO. A write to anywhere in the MDEU FIFO address space causes the 64-bit-words to be pushed onto the MDEU input FIFO, and a read from anywhere in the MDEU FIFO address space returns all zeros.

#### NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

### 12.4.5 Random Number Generator (RNG)

This section contains details about the random number generator (RNG), including modes of operation, status and control registers, and FIFOs.

The RNG is an execution unit capable of generating 64-bit random numbers. It is designed to comply with the FIPS-140 standard for randomness and non-determinism. A linear feedback shift register (LFSR) and cellular automata shift register (CASR) are operated in parallel to generate pseudo-random data.

The RNG consists of six major functional blocks:

- Bus interface unit (BIU)
- Linear feedback shift register (LFSR)
- Cellular automata shift register (CASR)
- Clock controller
- Six ring oscillators

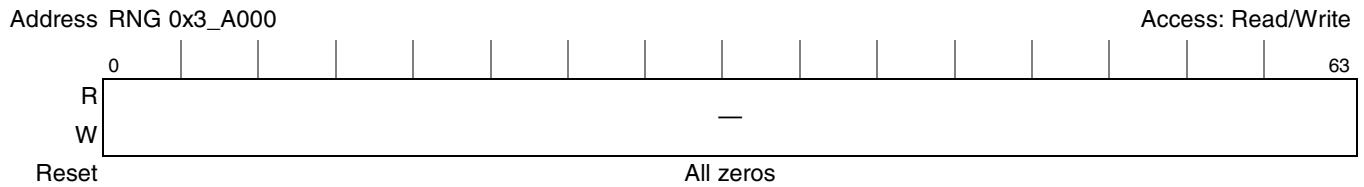
The states of the LFSR and CASR are advanced at unknown frequencies determined by the two ring oscillator clocks and the clock control. When a read is performed, the oscillator clocks are halted and a collection of bits from the LFSR and CASR are XORed together to obtain the 64-bit random output.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the MDEU is used through channel-controlled access, which means that most reads and writes of MDEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.



### 12.4.5.1 RNG Mode Register (RNGMR)

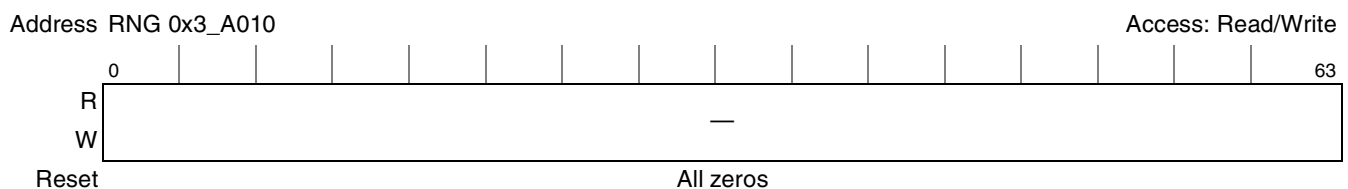
The RNG mode register is used to control the RNG. One operational mode, randomizing, is defined. The RNG mode register is a writable location but all mode bits are currently reserved. It is documented for the sake of consistency with the other EU's. The RNG mode register is shown in [Figure 12-40](#).



**Figure 12-40. RNG Mode Register**

### 12.4.5.2 RNG Data Size Register (RNGDSR)

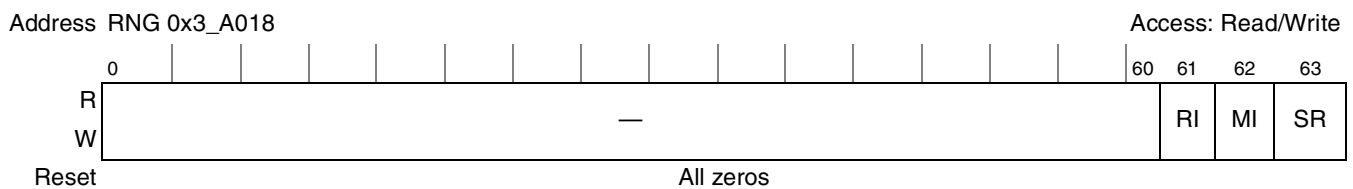
The RNG data size register is used to tell the RNG to begin generating random data. The actual contents of the data size register does not affect the operation of the RNG. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once the data size register is written, the RNG begins pushing data onto the FIFO. Data is pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG then attempts to keep the FIFO full.



**Figure 12-41. RNG Data Size Register**

### 12.4.5.3 RNG Reset Control Register (RNGRCR)

This register, shown in [Figure 12-42](#), contains three reset options specific to the RNG.



**Figure 12-42. RNG Reset Control Register**

Table 12-34 describes the RNG reset control register fields.

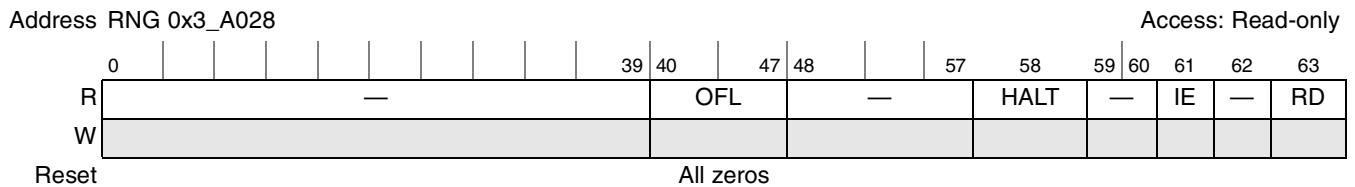
**Table 12-34. RNG Reset Control Register Field Descriptions**

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes RNG interrupts signaling DONE and ERROR to be reset. It further resets the state of the RNG interrupt status register. 0 No reset 1 Reset interrupt logic
62	MI	Module initialization. This reset value performs enough of a reset to prepare the RNG for another request, without forcing the internal control machines and the output FIFO to be reset, thereby invalidating stored random numbers or requiring reinvoation of a warm-up period. Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 No reset 1 Reset most of RNG
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for the RNG. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNG reset

#### 12.4.5.4 RNG Status Register (RNGSR)

This RNG status register, shown in Figure 12-43, contains 6 fields that reflect the state of the RNG internal signals.

The RNG status register is read-only. Writing to this location results in an address error being reflected in the RNG interrupt status register.



**Figure 12-43. RNG Status Register**

Table 12-35 describes the RNG status register fields.

**Table 12-35. RNG Status Register Field Descriptions**

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO.
48–57	—	Reserved
58	HALT	Halt. Indicates that the RNG has halted due to an error. 0 RNG not halted 1 RNG halted <b>Note:</b> Because the error causing the RNG to stop operating may be masked before reaching the interrupt status register, the RNG interrupt status register is used to provide a second source of information regarding errors preventing normal operation.

Table 12-35. RNG Status Register Field Descriptions (continued)

Bits	Name	Description
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR)”) 0 RNG is not signaling error 1 RNG is signaling error
62	—	Reserved
63	RD	Reset done. This status bit, when high, indicates that the RNG has completed its reset sequence. 0 Reset in progress 1 Reset done <b>Note:</b> Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

### 12.4.5.5 RNG Interrupt Status Register (RNGISR)

The RNG interrupt status register tracks the state of possible errors, if those errors are not masked, via the RNG interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 12-44.



Figure 12-44. RNG Interrupt Status Register

Table 12-36 describes RNG interrupt status register fields.

Table 12-36. RNG Interrupt Status Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error 0 No internal error detected 1 Internal error
52–55	—	Reserved
56	ME	Mode error. Indicates that the host has attempted to write an illegal value to the mode register 0 Valid data 1 Invalid data error
57	AE	Address error. An illegal read or write address was detected within the RNG address space. 0 No error detected 1 Address error
58–61	—	Reserved

Table 12-36. RNG Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
62	OFU	Output FIFO underflow. The RNG output FIFO was read while empty. 0 No overflow detected 1 Output FIFO has underflowed
63	—	Reserved

### 12.4.5.6 RNG Interrupt Control Register (RNGICR)

The RNG interrupt control register controls the result of detected errors. For a given error (as defined in Section 12.4.5.5, “RNG Interrupt Status Register (RNGISR)” ), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Address RNG 0x3\_A038

Access: Read-only

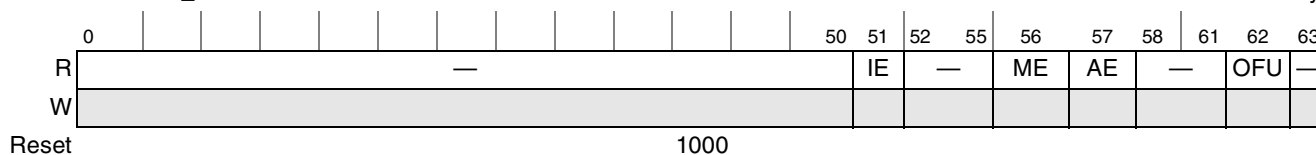


Figure 12-45. RNG Interrupt Control Register

Table 12-37 describes RNG interrupt status register fields.

Table 12-37. RNG Interrupt Control Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while generating random numbers. 0 Internal error enabled 1 Internal error disabled
52–55	—	Reserved
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–61	—	Reserved
62	OFU	Output FIFO underflow. RNG output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

### 12.4.5.7 RNG EU Go Register (RNGEUG)

The RNG EU go is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

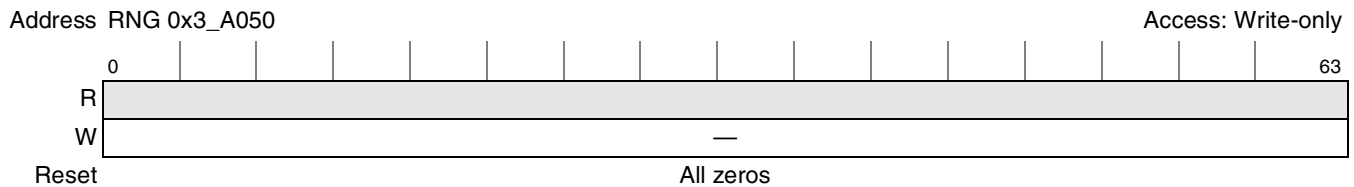


Figure 12-46. RNG EU Go Register

### 12.4.5.8 RNG FIFO

RNG uses an output FIFO to collect periodically sampled random 64-bit-words, with the intent that random data always be available for reading. The FIFO is multiple-addressed, but those multiple addresses point only to the appropriate end of the output FIFO. A read from anywhere in the RNG FIFO address space causes a 64-bit-word to be popped off of the RNG output FIFO. Underflows caused by reading or writing the RNG output FIFO are reflected in the RNG interrupt status register. Also, a write to the RNG output FIFO space will be reflected as an addressing error in the RNG interrupt status register.

## 12.4.6 Advanced Encryption Standard Execution Unit (AESU)

This section contains details about the advanced encryption standard execution unit (AESU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here are not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the ASEU is used through channel-controlled access, which means that most reads and writes of ASEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.

### 12.4.6.1 AESU Mode Register (AESUMR)

The AESU mode register, shown in [Figure 12-47](#), contains 7 bits which are used to program the AESU. The mode register is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

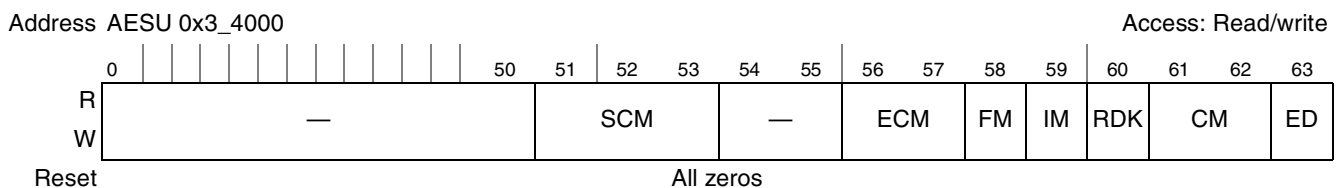


Figure 12-47. AESU Mode Register

Table 12-38 describes the AESU mode register fields.

**Table 12-38. AESU Mode Register**

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–50	—	Reserved
51–53	SCM	Sub-cipher-mode. Specifies additional options specific to particular cipher modes. <ul style="list-style-type: none"> <li>XOR cipher mode: specifies the number of sources to be XORed together. Valid values are 2 and 3.</li> <li>For all other cipher modes, this field must be 0.</li> </ul>
54–55	—	Reserved, must be set to zero.
The following bits are controlled through the MODE0 field of the descriptor header.		
56–57	ECM	Extend cipher mode. Used in combination with bits 61–62 (cipher mode) to define the mode of AES operation. See Table 12-39 for mode bit combinations.
58	FM	Final MAC (FM). Processes final message block and generates final MAC tag at end of message processing (CCM mode only) <ul style="list-style-type: none"> <li>0 Do not generate final MAC tag</li> <li>1 Generate final MAC tag after CCM processing is complete.</li> </ul>
59	IM	Initialize MAC(IM). Initializes AESU for new message (CCM mode only) <ul style="list-style-type: none"> <li>0 Do not initialize (context will be loaded by host)</li> <li>1 Initialize new message with nonce</li> </ul>
60	RDK	Restore decrypt key (RDK). Specifies that key data write will contain pre-expanded key (decrypt mode only). See note below on use of RDK bit. <ul style="list-style-type: none"> <li>0 Expand the user key prior to decrypting the first block</li> <li>1 Do not expand the key. The expanded decryption key will be written following the context switch.</li> </ul>
61–62	CM	Cipher mode. Used in combination with bits 56–57 (extend cipher mode) to define the mode of AES operation. See Table 12-39 for mode bit combinations.
63	ED	Encrypt/decrypt. If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. <ul style="list-style-type: none"> <li>0 Perform decryption</li> <li>1 Perform encryption</li> </ul> Note: This bit is ignored if CM is set to 0b11—CTR mode.

**Table 12-39. AES Cipher Modes**

Mode	ECM (56–57)	CM (61–62)
ECB	00	00
CBC	00	01
CTR	00	11
SRT <sup>1</sup>	01	11
CCM (without ICV comparison)	10	00
CCM with ICV comparison	11	00
XOR	11	11
Reserved	all others	

<sup>1</sup> SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010\_0 srtp. See Section 12.4.6.9.3, “Context for SRT Mode,” for more information on how SRT mode reduces context loading overhead.

## NOTE

Note on restore decrypt key (RDK)—In most networking applications, the decryption of an AES protected packet is performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.)

The use of RDK is completely optional, as the input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is 0100\_0- AESU key expand output. The AESU mode must be decrypted. See [Table 12-6](#) for more information. The descriptor causes the SEC to write the contents of the context registers and the key registers (containing the expanded decrypt key) to memory.

To process the remainder of the message, use a common descriptor type (0001\_0), and set the restore decrypt key mode bit. Load the context registers and the expanded decrypt key with previously saved key and context data from the first message. The key size is written as before (16, 24, or 32 bytes).

### 12.4.6.2 AESU Key Size Register (AESUKSR)

The AESU key size register stores the number of bytes in the key (16,24,32). Any key data beyond the number of bytes in the key size register will be ignored. This register is cleared when the AESU is reset or re-initialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error is generated. If the key size register is modified during processing, a context error is generated.



Figure 12-48. AESU Key Size Register

### 12.4.6.3 AESU Data Size Register (AESUDSR)

The AESU data size register, shown in [Figure 12-49](#), stores the number of bits in the final message block. Acceptable sizes vary depending on the AES mode selected. In ECB, CBC, and CTR mode, the message processed by the AESU must be a multiple of 128 bits; the AESU does not automatically pad messages out to 128-bit blocks. In CCM mode, data size must be a multiple of 8 bits. In XOR mode the data size

must be a multiple of 256 bits (32 bytes). If an improper data size is written, a data size error is generated. Only the lowest 3, 7, or 8 bits of the data size register are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

This register is cleared when the AESU is reset or re-initialized.

Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

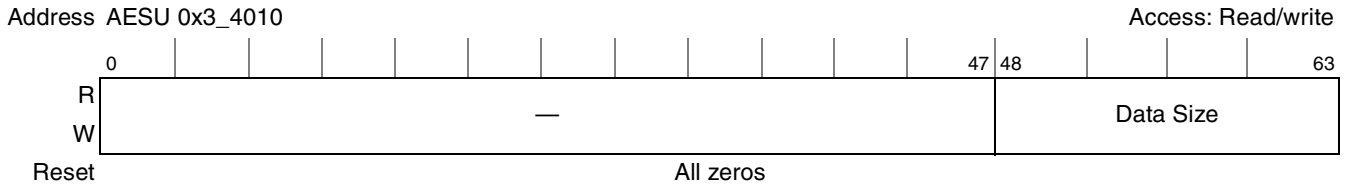


Figure 12-49. AESU Data Size Register

### 12.4.6.4 AESU Reset Control Register (AESURCR)

This register allows three levels of reset for the AESU, as defined by the three self-clearing bits:

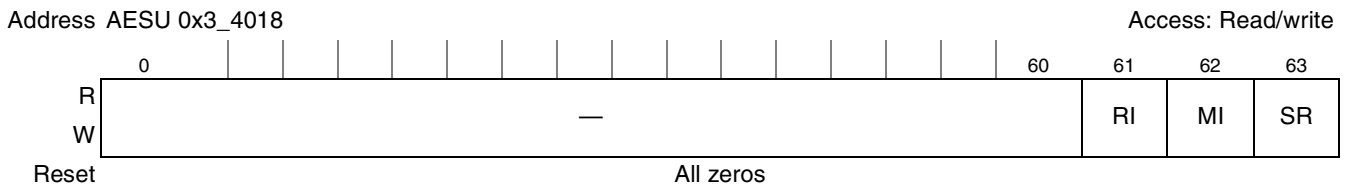


Figure 12-50. AESU Reset Control Register

Table 12-40 describes AESU reset control register fields.

Table 12-40. AESU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes AESU interrupts signaling DONE and ERROR to be reset. It further resets the state of the AESU interrupt status register. 0 Don't reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register 0 Don't reset 1 Reset most of AESU
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU enters a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register indicates when this initialization routine is complete 0 Don't reset 1 Full AESU reset



### 12.4.6.5 AESU Status Register (AESUSR)

AESU status register is a read-only register that reflects the state of six status outputs. Writing to this location will result in an address error being reflected in the AESU interrupt status register.

Address AESU 0x3\_4028

Access: Read-only

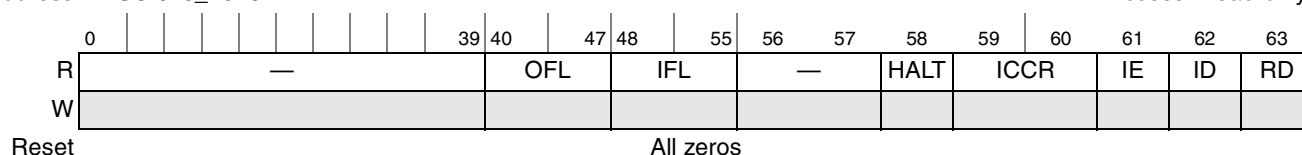


Figure 12-51. AESU Status Register

Table 12-41 describes the AESU status register fields.

Table 12-41. AESU Status Register Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the AESU has halted due to an error. 0 AESU not halted 1 AESU halted <b>Note:</b> Because the error causing the AESU to stop operating may be masked before reaching the interrupt status register, the AESU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved A passed or failed result is generated only if ICV checking is enabled and the cipher mode selected is CCM with ICV comparison
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR”). 0 AESU is not signaling error 1 AESU is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR”). 0 AESU is not signaling done 1 AESU is signaling done
63	RD	Reset done. This status bit, when high, indicates that AESU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done <b>Note:</b> Reset done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.



**Table 12-42. AESU Interrupt Status Register Field Descriptions (continued)**

Bits	Name	Description
58	OFE	Output FIFO error. The AESU output FIFO was detected non-empty upon write of AESU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The AESU input FIFO was detected non-empty upon generation of done interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO overflow. The AESU Input FIFO was pushed while full. 0 No error detected 1 Input FIFO has overflowed <b>Note:</b> When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the AESU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The AESU Output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

#### 12.4.6.7 AESU Interrupt Control Register (AESUICR)

The AESU interrupt control register, shown in Figure 12-53, controls the result of detected errors. For a given error (as defined in Section 12.4.6.6, “AESU Interrupt Status Register (AESUISR)”), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

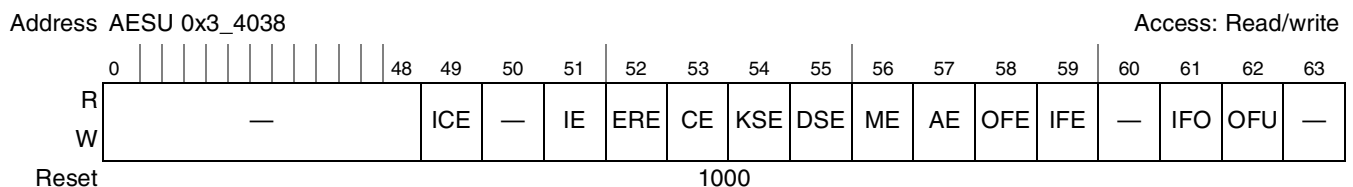
**Figure 12-53. AESU Interrupt Control Register**

Table 12-43 describes the AESU interrupt control register fields.

**Table 12-43. AESU Interrupt Control Register Field Descriptions**

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the AESU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved

**Table 12-43. AESU Interrupt Control Register Field Descriptions (continued)**

Bits	Name	Description
51	IE	Internal error. An internal processing error was detected while the AESU was processing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The AESU IV register was read while the AESU was processing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. An AESU key register, the key size register, data size register, mode register, or IV register was modified while the AESU was processing. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
58	OFE	Output FIFO error. The AESU output FIFO was detected non-empty upon write of AESU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The AESU input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The AESU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The AESU Output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

### 12.4.6.8 AESU EU Go Register (AESUEUG)

The AESU EU go register, shown in [Figure 12-54](#), is used to indicate an AES operation may be completed. After the final message block is written to the input FIFO, the EU go register must be written. The value in the data size register is used to determine how many bits of the final message block (always 128) will

be processed. Writing to this register causes the AESU to process the final block of a message, allowing it to signal DONE. A read of this register will always return a zero value.

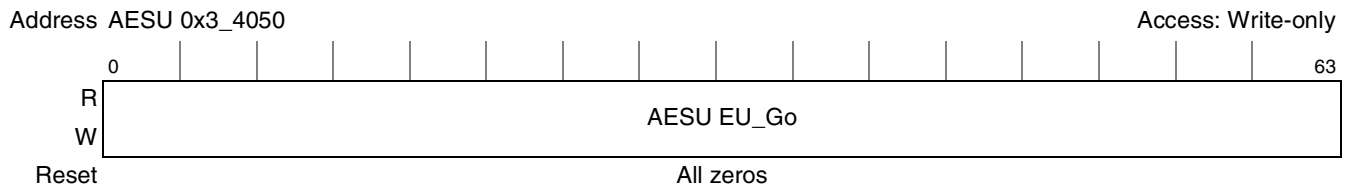


Figure 12-54. AESU EU Go Register

### 12.4.6.9 AESU Context Registers

There are seven 64-bit context data registers that allow the host to read/write the contents of the context used to process the message. The context must be written prior to the key data. If the context registers are written during message processing, a context error will be generated. All context registers are cleared when a hard/soft reset or initialization is performed.

The context registers must be read when changing context and restored to their original values to resume processing an interrupted message (CBC, CTR and CCM modes). For CTR and CCM mode, all seven 64-bit context registers must be read to retrieve context, and all seven must be written back to restore context. Effectively, the user must read the four empty place holder context registers in addition to the three context registers holding the counter and counter modulus exponent when in CTR mode. The contents of the empty context registers need not be preserved, but when restoring the CTR mode context, the 'empty' registers must be filled with 32 bytes of zeros before writing the saved counter and counter modulus exponent.

Context should be loaded with the lower bytes in the lowest 64-bit context register. The context registers are summarized in Figure 12-55.

Cipher Mode	Context Register (64 Bits Each)						
	1	2	3	4	5	6	7
ECB	—	—	—	—	—	—	—
CBC	IV1 <sup>1</sup>	IV2 <sup>1</sup>	—	—	—	—	—
CTR	—	—	—	—	Counter <sup>1</sup>		Counter modulus exponent <sup>1</sup>
SRT	Counter <sup>1</sup>		Counter modulus exponent (M) <sup>1</sup>	—	—	—	—
CCM	IV <sup>1</sup> /MAC Tag		Encrypted MAC <sup>2</sup> /decrypted MAC/encrypted counter		Counter <sup>1</sup>		Counter modulus exponent <sup>1</sup> /header size/MAC size <sup>3</sup>

Figure 12-55. AESU Context Register

- <sup>1</sup> Must be written at the start of a new message
- <sup>2</sup> Must be written at start of new CCM decryption
- <sup>3</sup> Header size/MAC size is only used if AES-CCM processing is suspended and resumed.

### 12.4.6.9.1 Context for CBC Mode

Within the context register, for use in CBC mode, are two 64-bit context data registers that allow the host to read/write the contents of the initialization vector (IV):

- IV1 holds the least significant bytes of the initialization vector (bytes 1–8).
- IV2 holds the most significant bytes of the initialization vector (bytes 9–16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the CBC mode bit is not set, a context error is generated.

The IV registers may only be read after processing has completed, as indicated by the setting of interrupt done (DONE) in the AESU status register as shown in [Section 12.4.6.5, “AESU Status Register \(AESUSR\)”](#). If the IV registers are read prior to assertion of interrupt done, an early read error is generated.

The IV registers must be read when changing context and restored to resume processing an interrupted message (CBC mode only).

### 12.4.6.9.2 Context for Counter Mode

In counter mode, a random 128-bit initial counter value is incremented modulo  $2^M$  with each block processed. The running counter is encrypted and exclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent  $M$  can be set between 8 and 128 in multiples of 8. The value of  $M$  is specified by writing to context register 3 as described in [Figure 12-56](#).

### 12.4.6.9.3 Context for SRT Mode

As was noted in the AESU mode register, SRT is not a new AES mode; it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010\_0 srtp. As with counter mode, a random 128-bit initial counter value is incremented modulo  $2^M$  with each block processed. The running counter is encrypted and exclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent  $M$  can be set between 8 and 128 in multiples of 8. The value of  $M$  is specified by writing to context register 3 as described in [Figure 12-56](#).

The only difference between SRT mode and CTR mode is that in SRT mode, the AES context is loaded and read via context registers 1–3, with no requirement to access context registers 4–7. In CTR mode, context registers 1–4 must be loaded with zeros, with the counter and modulus being loaded into and read from context registers 5–7.

### 12.4.6.9.4 Context for CCM Mode

The SEC AESU is capable of performing single pass encryption and MAC generation. The host is required to order the CCM context in such a way that the context can be fetched as a contiguous string into the context registers, prior to encryption/MAC generation or decryption/MAC validation. The context register contents for CCM mode is summarized in Figure 12-56 and further described below.

		Context Registers						
		1	2	3	4	5	6	7
Encrypt (outbound)	Inputs	IV		0		Initial Counter		Counter Modulus Exponent
	Outputs	MAC	0	MIC	0			
Decrypt (inbound)	Inputs	IV		MIC	0	Initial Counter		Counter Modulus Exponent
	Outputs	Computed MAC	0	Decrypted MAC	0			

**Figure 12-56. AESU CCM Context Registers**

The context for CCM encryption/MAC generation is:

- Reg 1–2 Session-specific 128-bit initialization vector (from memory)
- Reg 3–4 128 bits of zero padding
- Reg 5–6 Session-specific counter (initial counter value) (from memory)
- Reg 7 Counter modulus exponent (msb<--lsb); should be fixed at 0x0000\_0080.

Note: The counter modulus for CCM mode is currently defined as  $2^{128}$ , making the exponent 128. This value has been made programmable in the SEC in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session-specific information for loading into the AESU context register prior to CCM encryption.

#### CCM encryption processing

With the session-specific key and context, the AESU performs the following operations.

1. Initialize the IV, and encrypt with the symmetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symmetric key.
3. Continue as in step 2 until the final block of plaintext has been processed. The result of the encryption of the final block of plaintext with the symmetric key is the MAC tag. The full 128 bits of MAC data is written to context registers 1–2, for use in the next phase of CCM processing. Once the MAC Tag has been generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in counter mode.
4. The first item to be encrypted in counter mode is the counter (initial counter value) from context registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC tag (retrieved from context register 1–2) to produce the MIC (encrypted MAC), which is then stored in context registers 3–4. At the completion of CCM encrypt processing, this MIC is output

to memory (per the descriptor pointer) for the host to append to the 802.11i frame. Note: The MIC written out to memory by the AESU is the full 128 bits. The host must only append the most significant 64 bits to the frame as the MIC.

5. The counter value is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of plaintext to produce the first block of cipher text. The ciphertext is placed in the AESU output FIFO.
6. The counter continues to be incremented, and encrypted with the symmetric key, with the result hashed with each successive block of plaintext, until all plaintext has been converted to ciphertext. The SEC controller will manage FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the MIC has been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is:

- Reg 1–2 Session-specific 128-bit initialization vector (from memory)
- Reg 3–4 MIC (from received frame) + 64 bits of zero padding
- Reg 5–6 Session-specific counter (initial counter value) (from memory)
- Reg 7 Counter modulus exponent (msb<--lsb) Should be fixed at 0x0000\_0080.

#### NOTE

The counter modulus for CCM mode is currently defined as  $2^{128}$ , making the exponent 128. This value has been made programmable in the SEC to in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU context register prior to CCM decryption.

CCM decryption processing is the reverse of encryption. With the session specific key and context, the AESU performs the following operations:

1. Initialize the IV, and encrypt with the symmetric key. Simultaneously, the counter (initial counter value) from context registers 5–6 is encrypted with the symmetric key. The result is hashed with the encrypted MAC (from context register 3–4), and the resulting original MAC is written to context register 3–4, overwriting the encrypted MAC.

**Note:** Strictly speaking, the counter is encrypted with the symmetric key; however, the AESU should be set for decrypt to perform the counter and CBC processes in the correct order.

2. The 802.11 frame header is hashed with the encrypted IV. (The AESU automatically determines the header length.) Simultaneously, the counter is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of ciphertext to produce the first block of plaintext. The plaintext is placed in the AESU output FIFO, while simultaneously, in CBC fashion, a copy of the first block of plaintext is hashed with the output of encryption of the 802.11 frame header. The output is encrypted with the symmetric key.
3. As each ciphertext block is converted to plaintext, the plaintext is CBC encrypted. When the final plaintext block has been processed, the CBC MAC (MAC tag) is written to context registers 1-2. The first 64 bits of the MAC tag are compared to the MAC tag recovered in step 1.



## NOTE

For both encrypt and decrypt operations, if the 802.11 frame is being processed as a whole (not split across multiple descriptors), the initialize and final MAC bits should be set in the AESU mode register.

### 12.4.6.9.5 AESU Key Registers

The AESU key registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to key 1. Any key data written to bytes beyond the value written to the key size register is ignored. The key data registers are cleared when the AESU is reset or re-initialized. If these registers are modified during message processing, a context error is generated.

The key data registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the restore decrypt key bit must be set in the mode register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

### 12.4.6.9.6 AESU FIFOs

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted with the key and initialization vector (CBC mode only) and the results are placed in the output FIFO. The output size is the same as the input size.

Writing to the FIFO address space places 64 bits of message data into the input FIFO. The input FIFO may be written any time the number of dwords currently in the input FIFO (as indicated by the IFL field of the AESU status register) is less than 32. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the data size register.

Reading from the FIFO address space will pop 64 bits of message data from the output FIFO. The output FIFO may be read any time the OFR signal is asserted (as indicated in the AESU status register). This indicates that the number of bytes in the output FIFO is at or above the threshold specified in the mode register.

## 12.4.7 Kasumi Execution Unit (KEU)

This section contains details about the Kasumi execution unit (KEU), including modes of operation, status and control registers, and FIFOs. The Kasumi execution unit (KEU) has been designed to support the F8 confidentiality function of the 3GPP, GSM A5/3, EDGE A5/3 and GPRS GEA3 algorithms and also the 3GPP F9 integrity function.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the KEU is used through channel-controlled access, which means that most reads and writes of KEU registers are directed by the SEC channels. Driver software performs host-controlled register accesses on only a few registers for initial configuration and error handling.

### 12.4.7.1 KEU Mode Register (KEUMR)

The KEU mode register, shown in Figure 12-57, contains several bits which are used to program the KEU. The mode register is cleared when the KEU is reset or re-initialized. Setting a reserved mode bit generates a data error. Setting both the GSM and EDGE bits to one generates a data error. If the KEU mode register is modified during processing, a context error is generated.

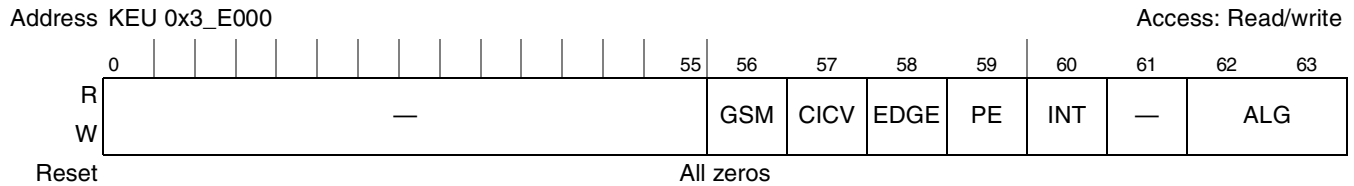


Figure 12-57. KEU Mode Register

Table 12-44 describes the KEU mode register fields.

Table 12-44. KEU Mode Register Field Descriptions

Bits	Name	Description
0–55	—	Reserved
56	GSM	Select GSM A5/3 blocks 0 GSM A5/3 blocks not selected 1 GSM A5/3 blocks selected <b>Note 1:</b> For GSM A5/3, Two 114-bit blocks are required to be produced each 4.615mS slot. If GSM = 1, the first read of the output FIFO retrieves the first 64 bits of block 1. The second read of the output FIFO retrieves the next 50 bits of block 1 (the remaining bits of this 64-bit word are set to zero). The third read of the output FIFO retrieves the first 64 bits of block 2, while a fourth read of the output FIFO retrieves the next 50 bits of block 2 (the remaining bits of this 64-bit word are set to zero). <b>Note 2:</b> If GSM = 0, 228 contiguous bits may be read with successive reads of the output FIFO. In this case the host (application) will be responsible for handling the A5/3 block formatting. <b>Note 3:</b> If GSM is set to 1, while EDGE = 1, an interrupt/error will be generated.
57	CICV	Compare integrity check values. 0 Normal operation; no ICV comparison. 1 After the ICV is computed, compare it to the data in the KEU's ICV_In register. If the ICVs do not match, send an error interrupt to the channel. Only applicable to
58	EDGE	Select EDGE A5/3 blocks 0 EDGE A5/3 blocks not selected 1 EDGE A5/3 blocks selected <b>Note 1:</b> For EDGE A5/3, Two 348-bit blocks are required to be produced each 4.615mS slot. If EDGE = 1, the first five reads of the output FIFO retrieve the first 320 bits of block 1. The sixth read of the output FIFO retrieves the final 28 bits of block 1 (the remaining bits of the sixth 64-bit word are set to zero). The next five reads of the output FIFO retrieve the first 320 bits of block 2. The following read of the output FIFO retrieves the final 28 bits of block 2 (the remaining bits of this 64-bit word are set to zero). <b>Note 2:</b> If EDGE = 0, 696 contiguous bits may be read with successive reads of the output FIFO. In this case the host (application) will be responsible for handling the A5/3 block formatting. <b>Note 3:</b> If EDGE is set to 1, whilst GSM = 1, an interrupt/error will be generated.

Table 12-44. KEU Mode Register Field Descriptions (continued)

Bits	Name	Description
59	PE	Process end of message. Enables final processing of last message block (F9 only). 0 Prevent final block processing (message incomplete) 1 Enable final block processing (message complete) <b>Note:</b> For f9 operations, if the 3G frame (or message) is being processed as a whole (not split across multiple descriptors), the process end of message bit should be set. If the frame is processed across multiple descriptors, this bit should only be set on the descriptor performing f9 processing on the final message block.
60	INT	Initialization. Enables initialization for a new message. 0 Prevent initialization 1 Enable initialization <b>Note:</b> For F8 or F9 operations, if the 3G frame (or message) is being processed through a single descriptor, the Initialization bit should be set. If the frame is split across multiple descriptors, this bit should only be set in the descriptor that processes the first block of the message.
61	—	Reserved
62–63	ALG	Algorithm selection. Specifies the functions to perform. 00 Perform F8 function only 01 Reserved 10 F9 function only 11 Reserved

### 12.4.7.2 KEU Key Size Register (KEUKSR)

The KEU key size register, shown in [Figure 12-58](#), stores the number of bytes in the key. It should be set to 16 bytes. This register is cleared when the KEU is reset or re-initialized. If a key size is specified that does not match the selected algorithm(s), an illegal key size error is generated.

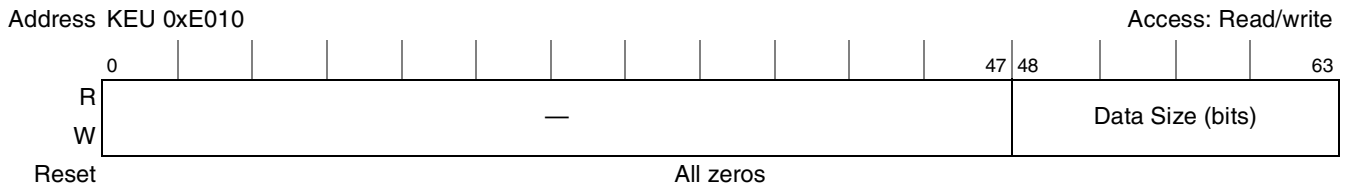


Figure 12-58. KEU Key Size Register

### 12.4.7.3 KEU Data Size Register (KEUDSR)

The KEU data size register stores the number of bits to process in the final message word. Because Kasumi allows for bit level granularity for encryption/decryption, there are no illegal data sizes. The proper bit length of the message must be written to notify the KEU of any padding performed by the host. This register is cleared when the KEU is reset or re-initialized.

Writing to this register signals the KEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.



**Figure 12-59. KEU Data Size Register**

For the F8 function, for example, if the 64-bit F8 keystream is 0x1234567890ABCDEF and the data size register contains 0x0a (10 = 1 byte + 2 bits), the final ten message bits will be exclusive-ORed (XORed) with ten bits of keystream 0x120. The PE (process end of message) mode bit must be set.

For the 3GPP F9 function: The final 64 bits of the message will be padded as specified in the 3GPP F9 algorithm. The PE (process end of message) mode bit must be set.

3GPP F9 padding is automatically performed. The CD (communication direction) bit (see Table 12-49) and 1 is appended to the end of the message. The result is then zero-padded to 64 bits.

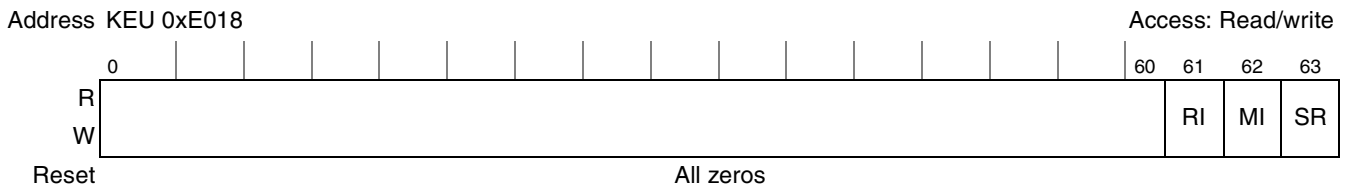
For example, if the last block is xF81A\_0000\_0000\_0000 (big endian) and data size contains 0x0F (15 bits = 1 byte + 7 bits), the word (0xF81A = 1111\_1000\_0001\_1010) the most-significant 15 bits (underlined) are padded left to right as follows:

1111 1000 0001 101\$\_1000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000

where \$ is the value of the CD (communication direction) bit in the mode register.

### 12.4.7.4 KEU Reset Control Register (KEURCR)

The KEU reset control register, shown in Figure 12-60, allows 3 levels of reset of the KEU, as defined by the 3 self-clearing bits:



**Figure 12-60. KEU Reset Control Register**

Table 12-45 describes the KEU reset control register fields.

**Table 12-45. KEU Reset Control Register Field Descriptions**

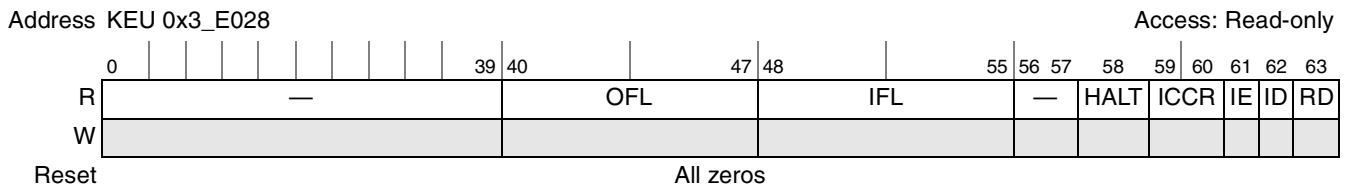
Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes KEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the KEU interrupt status register. 0 Don't reset 1 Reset interrupt logic

**Table 12-45. KEU Reset Control Register Field Descriptions (continued)**

Bits	Name	Description
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the KEU status register 0 Don't reset 1 Reset most of KEU
63	SR	Software reset. Functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for KEU. All registers and internal state are returned to their defined reset state. Upon negation of the SR bit, the KEU enters a routine to perform proper initialization of the parameter memories. The RD (reset done) bit in the KEU status register indicates when this initialization is complete 0 Normal operation 1 Full KEU reset

### 12.4.7.5 KEU Status Register (KEUSR)

The KEU status register, shown in [Figure 12-61](#), is a read-only register that reflects the state of six status outputs. Writing to this location results in an address error being reflected in the KEU interrupt status register.

**Figure 12-61. KEU Status Register**

[Table 12-46](#) describes the KEU status register fields.

**Table 12-46. KEU Status Register Field Descriptions**

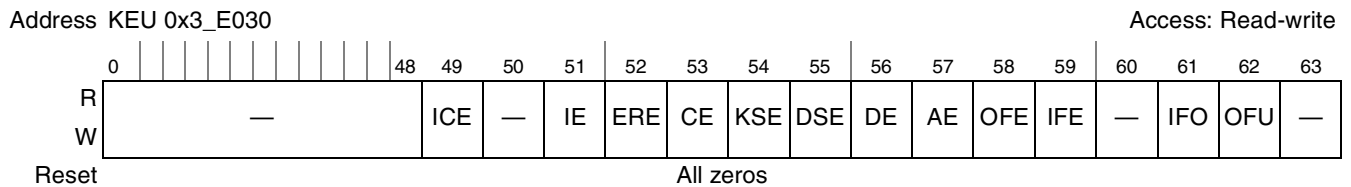
Bits	Name	Description
0–39	—	Reserved
40–47	OFL	Output FIFO level. The number of dwords currently in the output FIFO
48–55	IFL	Input FIFO level. The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Indicates when the KEU core has halted due to an error. 0 KEU not halted 1 KEU core halted (must be reset/re-initialized) <b>Note:</b> Because the error causing the KEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.

**Table 12-46. KEU Status Register Field Descriptions (continued)**

Bits	Name	Description
59-60	ICCR	Integrity check comparison result. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved <b>Note:</b> A passed or failed result is generated only if ICV checking is enabled and the algorithm selected is F9.
61	IE	Interrupt error. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR)”). 0 KEU is not signaling error 1 KEU is signaling error
62	ID	Interrupt done. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 12.6.5.3, “Interrupt Status Register (ISR)”). 0 Processing not done 1 All bytes processed
63	RD	Reset done. Indicates when the KEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done

### 12.4.7.6 KEU Interrupt Status Register (KEUISR)

The KEU interrupt status register tracks the state of possible errors, provided those errors are not masked, via the KEU interrupt control register. The definition of each bit in the KEU interrupt status register is shown in Figure 12-62.



**Figure 12-62. KEU Interrupt Status Register**

Table 12-47 describes the KEU interrupt status register signals.

**Table 12-47. KEU Interrupt Status Register Field Descriptions**

Bits	Signal	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 No error detected 1 Integrity check error detected. An ICV check was performed on an F9 result and the supplied ICV did not match the one computed by the KEU.
50	—	Reserved

Table 12-47. KEU Interrupt Status Register Field Descriptions (continued)

Bits	Signal	Description
51	IE	Internal error. An internal processing error was detected while the KEU was processing. 0 No error detected 1 Internal error <b>Note:</b> This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the interrupt control register or by resetting the KEU.
52	ERE	Early read error. A KEU context or IV register was read while the KEU was processing. 0 No error detected 1 Early read error
53	CE	Context error. A KEU key register, the key size register, the data size register, the mode register, or IV register was modified while the KEU was processing 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (not 16 or 32) was written to the KEU key size register 0 No error detected 1 Key size error
55	DSE	Data size error. A value was written to the KEU data size register that is greater than 64 bits. 0 No error detected 1 Data size error
56	DE	Data error. Invalid data was written to a register or a reserved mode bit was set. 0 Valid data 1 Reserved or invalid mode selected
57	AE	Address error. An illegal read or write address was detected within the KEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The KEU output FIFO was non-empty upon write of KEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The KEU input FIFO was non-empty upon generation of done interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO overflow. The KEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed
62	OFU	Output FIFO underflow. The KEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

### 12.4.7.7 KEU Interrupt Control Register (KEUICR)

The KEU interrupt control register controls the result of detected errors. For a given error (as defined in [Section 12.4.7.6, “KEU Interrupt Status Register \(KEUISR\)”](#)), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the KEU interrupt status register is not updated

to reflect the error. If the corresponding bit is not set, then upon detection of an error, the KEU interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing. The definition of each bit in the KEU interrupt control register is shown in Figure 12-63.

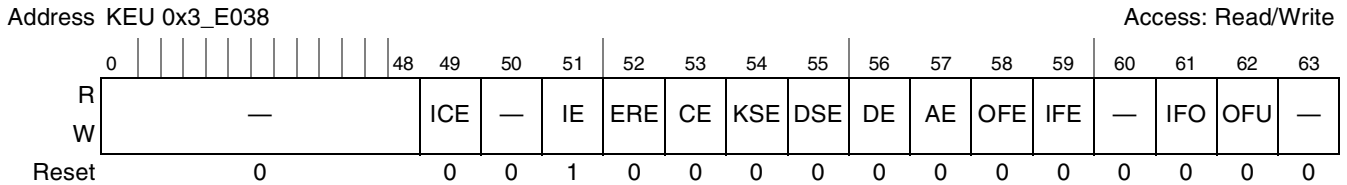


Figure 12-63. KEU Interrupt Control Register

Table 12-48 describes the KEU interrupt control register fields.

Table 12-48. KEU Interrupt Control Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 ICV check error enabled 1 ICV check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. A KEU context or IV register was read while the KEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. A KEU key register, the key size register, data size register, mode register, or IV register was modified while KEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (not 16 or 32 bytes) was written to the KEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	DE	Data error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Data error enabled 1 Data error disabled
57	AE	Address error. An illegal read or write address was detected within the KEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The KEU output FIFO was detected non-empty upon write of KEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled



Table 12-48. KEU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
59	IFE	Input FIFO error. The KEU input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The KEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The KEU output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

### 12.4.7.8 KEU Data Out Register (F9 MAC) (KEUDOR)

Following a done interrupt, the read-only KEU data out register holds the F9 message authentication code. A 64-bit value is returned. This value may be truncated to 32 bits for some applications. Writing to this location results in an address error being reflected in the KEU interrupt status register.

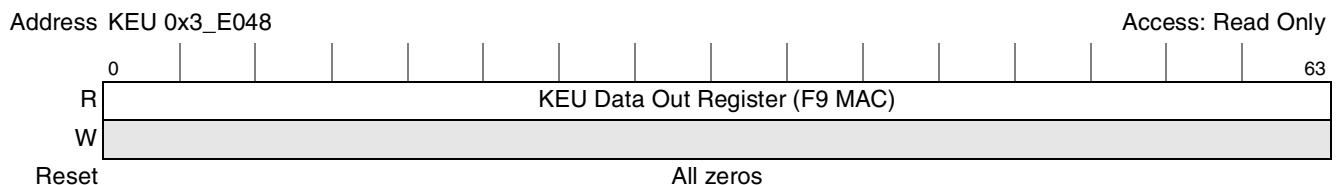


Figure 12-64. KEU Data Out Register (F9 MAC)

#### NOTE

According to the ETSI/SAGE 3GPP specification for F9 (version 1.2), only 32 bits of the final MAC are used. This corresponds to the lower 4 bytes of the KEU data out register.

### 12.4.7.9 KEU EU Go Register (KEUEUG)

The KEU EU go register, shown in [Figure 12-65](#), is used to signal to the KEU that the final message block has been written to the input FIFO. Writing to this register causes the KEU to process the final block of a message, allowing it to signal DONE. When processing the last block, the value in the data size register determines how many bits of the final message word (1–64) are processed. The value written to this register does not matter. A read of this register always returns a zero value.

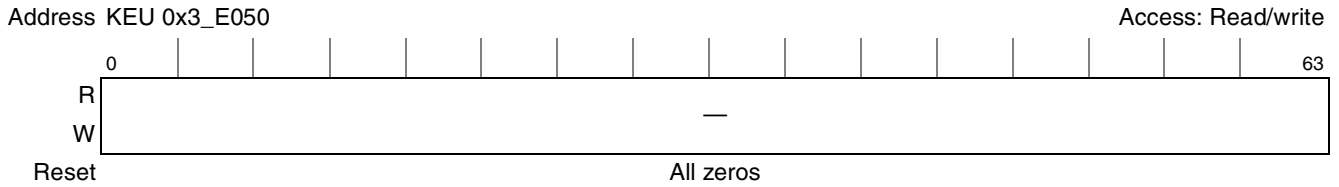


Figure 12-65. KEU EU Go Register

### 12.4.7.10 KEU IV 1 Register (KEUIV1)

The KEUIV1 register is a general purpose IV register used during the initialization phase of the F8 algorithms for 3GPP, GSM A5/3, EDGE A5/3, GPRS GEA3 and also for the F9 algorithm for 3GPP. The appropriate value as defined by the standards for each algorithm must be written before a new message is started. After the initialization phase has been completed, KEUIV1 register is no longer used for the remainder of F8 or F9 processing. However, if 3GPP F9, is selected, because the KEUIV1 register contains the direction bit as defined by the 3GPP standard, the KEUIV1 register must be written back during context switches, to complete the generation of the 3GPP MAC.

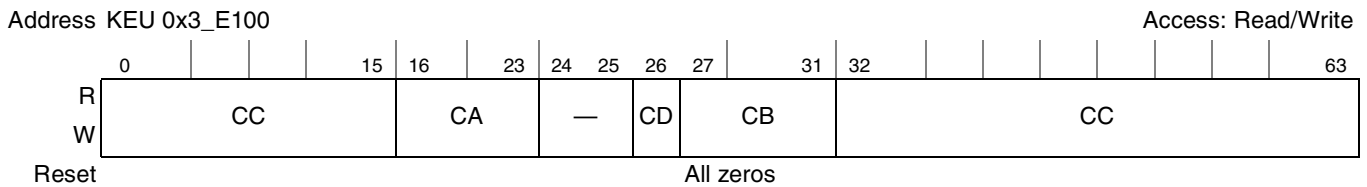


Figure 12-66. KEU IV\_1 Register

Table 12-49. KEU IV\_1 Register Field Descriptions

Bits	Name	Description
0–15	CC	Defined by the algorithm selected
16–23	CA	Defined by the algorithm selected
24–25	0	Reserved
26	CD	Defined by the algorithm selected (For 3GPP and GPRS, this is the communication direction bit)
27–31	CB	Defined by the algorithm selected
31–63	CC	Defined by the algorithm selected

#### NOTE

It is the responsibility of the user to ensure that fields of the IV 1 register are programmed correctly in accordance with the algorithm selected.

### 12.4.7.11 KEU ICV\_In Register (KEUICV)

If ICV checking is required, then the value to be compared with the computed F9 MAC value must be written to the ICV\_In register before data size is written.

### 12.4.7.12 KEU IV\_2 Register (Fresh) (KEUIV2)

The fresh value, KEU IV\_2 register, is used during the initialization phase of the 3GPP F9 algorithm. This value is ignored when the F8 algorithm is selected. The fresh value must be written before a new message to be processed with 3GPP F9 is started. After the initialization phase has been completed, KEU IV\_2 register is no longer used during message processing. KEU IV\_2 register need not be written during context switches.

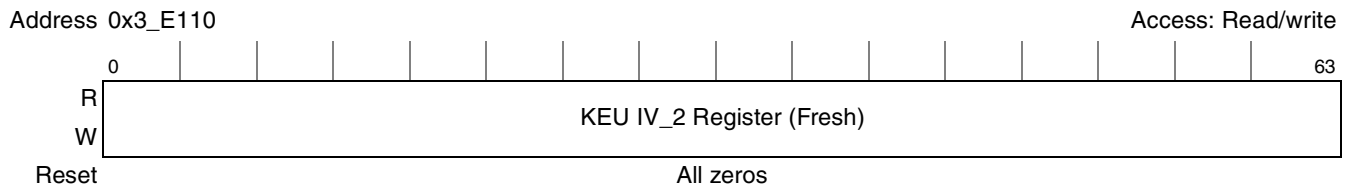


Figure 12-67. KEU IV\_2 Register (Fresh)

### 12.4.7.13 KEU Context Data Registers (KEUC<sub>n</sub>)

There are 6, 64-bit KEU context data registers that allow the host to read/write the contents of the context used to process the message. The KEU context data registers must be read when changing context and restored to their original values to resume processing an interrupted message. For F8 and 3GPP F9 modes, all 6 64-bit KEU context data registers must be read to retrieve context, and all 6 must be written back to restore context. The context must be written prior to the key data. If any of the KEU context data registers are written during message processing, a context error will be generated. All KEU context data registers are cleared when a hard/soft reset or initialization is performed.

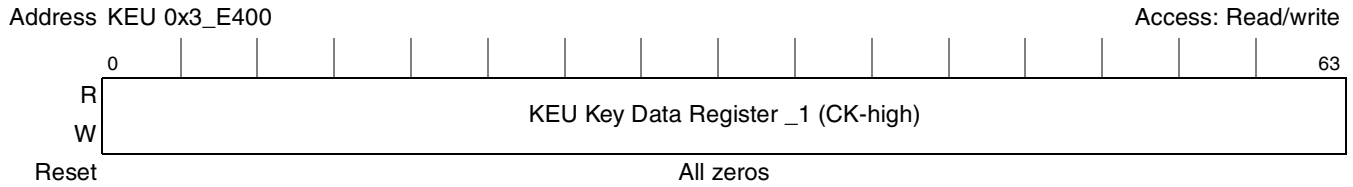
#### NOTE

In typical operation, a frame will be received and processed in its entirety, with the KEU performing session specific initialization using the contexts of KEU IV\_1 and IV\_2 registers. The KEU context data and IV\_1 registers should only be unloaded/reloaded when the processing of a frame is discontinued prior to completion; then processing is resumed.

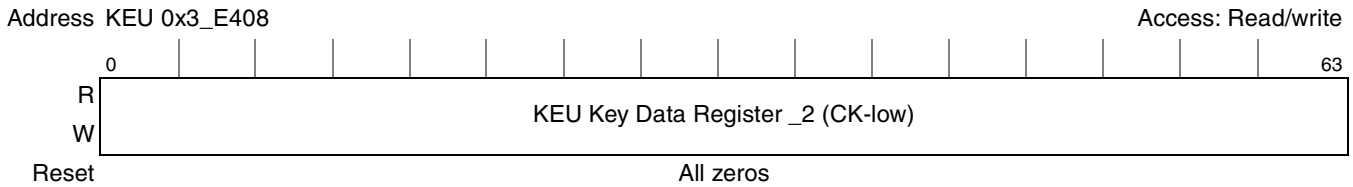
### 12.4.7.14 KEU Key Data Registers\_1 and \_2 (Confidentiality Key) (KEUKD<sub>n</sub>)

The first two KEU key data registers together hold one 128-bit key that is used for F8 encryption/decryption. KEU key data register\_1, (CK-high), holds the first 8 bytes (1-8). KEU key data register\_2, (CK-low), holds the second 8 bytes (9-16). The KEU key data registers must be written before message processing begins and cannot be written while the block is processing data, or a context error occurs.

Reading from either of these registers results in an address error being reflected in the KEU interrupt status register.



**Figure 12-68. KEU Key Data Register \_1 (CK-high)**



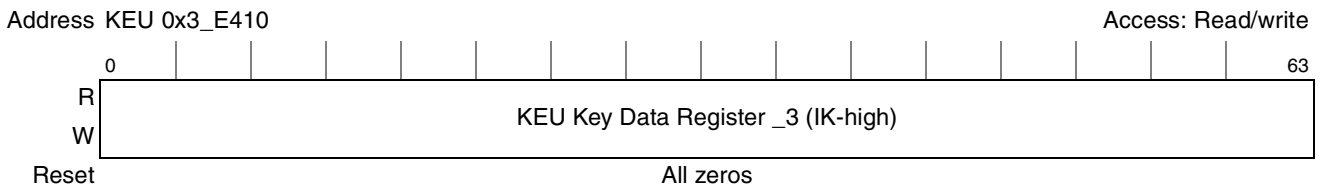
**Figure 12-69. KEU Key Data Register \_2 (CK-Low)**

### 12.4.7.15 KEU Key Data Registers \_3 and \_4 (Integrity Key) (KEUKD<sub>n</sub>)

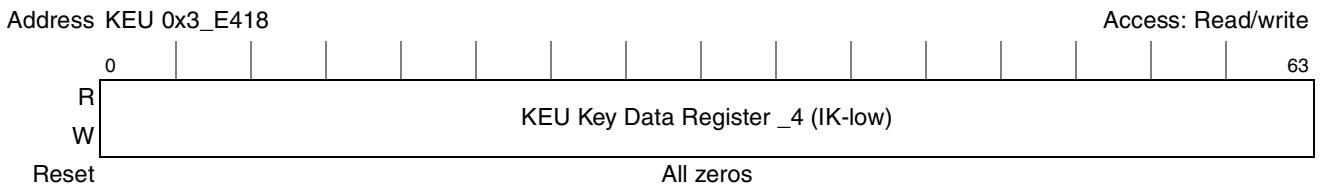
The KEU key data registers\_3 & \_4 together hold one 128-bit key that is used for F9 message authentication. KEU key data register\_3, (IK-high), holds the first 8 bytes (1-8). KEU key data register\_4, (IK-low), holds the second 8 bytes (9-16). The KEU key data registers must be written before message processing begins and cannot be written while the block is processing data, or a context error will occur.

If the F9 only mode is set, the integrity key data may be optionally written to KEU key data registers\_1 & \_2. This eliminates the need for the host to offset from the base key address to write to KEU key data registers\_3 & \_4 while using the KEU exclusively for the F9 integrity function.

Reading from either of these registers will result in an address error being reflected in the KEU interrupt status register.



**Figure 12-70. KEU Key Data Register \_3 (IK-high)**



**Figure 12-71. KEU Key Data Register \_4 (IK-low)**

### 12.4.7.16 KEU FIFOs

The KEU fetches data 64 bits at a time from the KEU input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the KEU output FIFO. During F9

processing, the input data is hashed with the integrity key and the resulting MAC is placed in the KEU data out register. The output size is the same as the input size.

A write to anywhere in the KEU FIFO address space causes the 64-bit-word to be pushed onto the KEU input FIFO, and a read from anywhere in the KEU FIFO address space causes a 64-bit-word to be popped off of the KEU output FIFO. Overflows and underflows caused by reading or writing the KEU FIFOs are reflected in the KEU interrupt status register.

## 12.5 Crypto-Channels

A channel in the SEC manages the execution of each cryptographic task, making use of one or more of the SEC's execution units (EUs). Control information and data pointers for a given task are stored in the form of a descriptor (see [Section 12.3.1, "Descriptor Structure"](#)) in system memory or in the channel itself. A descriptor determines what EUs are used, how they are configured, where to fetch needed data, and where to store the results. To invoke cryptographic tasks, the host constructs a descriptor, selects a channel, and writes a pointer to the descriptor into the selected channel's fetch FIFO. The fetch FIFO can store up to 24 pointers. Operations performed by channels include the following (not necessarily in this order):

- If the channel is idle and its fetch FIFO is non-empty, read the next descriptor pointer from the fetch FIFO, and use this pointer to read the descriptor into the channel's descriptor buffer.
- Request from the controller the assignment of one or more EUs for the exclusive use of the channel. Where necessary, configure the secondary EU to snoop input or output data intended for the primary EU.
- Upon notification of completion of the EU reset sequence, initialize mode registers in the assigned EU.
- Initialize EUs and write to EU registers such as key size and text-data size.
- Transfer data parcels (up to 32 Kbytes) from system memory into assigned EU input registers and FIFOs. This may involve using link tables to gather input data that has been split into multiple segments which are stored in various locations of system memory. For the RAID-XOR descriptor type, the channel rotates among three data sources, fetching 32 bytes from each source.
- Transfer data parcels (up to 32 Kbytes) from assigned EU output registers and FIFOs to system memory space. This may involve using link tables to scatter output data into multiple segments which are stored in various locations of system memory.
- Initialize the EU go register (where applicable) in the assigned EU upon completion of last EU write indicated by the descriptor. The channel waits for a indication from the EU that processing of input text-data is complete before proceeding with further activity after writing EU go.
- Reset assigned EU(s).
- Release assigned EU(s).
- When a descriptor has been completely processed, provide feedback to the host, in the form of interrupt and/or descriptor header write-back to system memory.
- When descriptor processing is halted due to an error, provide feedback to the host via interrupt.

The channel waits indefinitely for the controller to complete a requested activity before continuing to the next step of descriptor processing.

The channel can generate two types of done notification signals when it completes operation on a descriptor—an interrupt and/or a writeback of the descriptor header. The done interrupt is enabled by the CDIE bit and the writeback is enabled by the CDWE bit of the channel configuration register (Table 12-50). Table 12-51 shows the DONE field that is written back in if writeback is enabled.

The selected done notification can be performed at the end of processing of every descriptor, or only on selected descriptors. If the NT field is 0 in the channel configuration register, then done notification is performed after every descriptor. If the NT field is 1, then done notification is only performed on descriptors in which the DN bit is set in the packet header (Table 12-4).

### 12.5.1 Channel Registers

The SEC includes four channels that manage data and EU function. Each channel contains the following:

- A fetch FIFO, which holds a queue of pointers to descriptors waiting to be serviced
- A configuration register, which allows the user a number of options for SEC event signaling.
- Control registers containing information about the transaction in process
- A status register containing an indication of the last unfulfilled bus request
- A descriptor buffer memory used to store the active descriptor
- Scatter and gather link table buffer memory used to store the active link table

These registers are described in detail below.

#### 12.5.1.1 Crypto-Channel Configuration Registers 1–4 (CCCR<sub>n</sub>)

These register contains nine operational bits permitting configuration of the channel as shown in Figure 12-72.

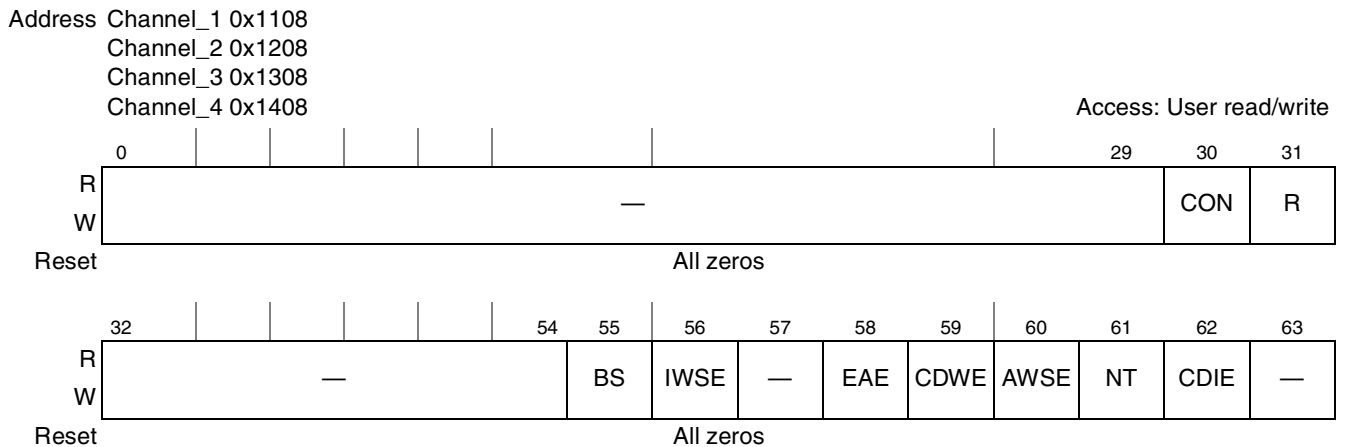


Figure 12-72. Crypto-Channel Configuration Register (CCCR)

Table 12-50 describes the CCCRs.

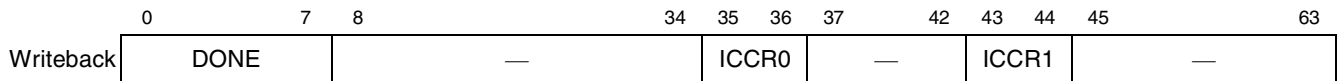
**Table 12-50. Crypto-Channel Configuration Register (CCCR) Field Descriptions**

Bits	Name	Description
0–29	—	Reserved. Set to zero.
30	CON	Continue. 0 No special action. 1 Causes the same channel reset actions as bit R, except that the fetch FIFO and the lower half of the CCR register are not cleared. After the reset sequence is complete, this bit automatically returns to 0 and the channel resumes normal operation, servicing the next descriptor pointer in the fetch FIFO, if any.
31	R	Reset channel. 0 No special action. 1 Causes a software reset of the channel, clearing all its internal state. The details of the software reset actions depend upon what the channel is doing when the bit is set: If the R bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request by asserting the release output signals. The channel then resets all its registers, clears the R bit, and returns the channel state machine to the idle state. If the R bit is set after the channel has been assigned an EU, the channel requests a write from the controller to set the software reset bit of the EU. If a secondary EU has been reserved, the channel requests a write to reset that EU as well. The channel next asserts the appropriate release signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the RESET bit and returns the channel state machine to the idle state.
32–54	—	Reserved. Set to zero.
55	BS	Burst size. The SEC accesses long text-data parcels in main memory through bursts of programmable size: 0 Burst size is 64 bytes 1 Burst size is 128 bytes
56	IWSE	ICV writeback status enable. 0 No special action. 1 If the descriptor calls for ICV comparison, then at the completion of descriptor processing, write back the status of primary and secondary EUs into the header dword.
57	—	Reserved. Set to zero.
58	EAE	Extend address enable. This bit determines whether the channel uses a 36-bit address bus or a 32-bit address bus. 0 Channel's address bus is 32 bits. 1 Channel's address bus is 36 bits. (Not available in SEC 2.1)
59	CDWE	Channel done writeback enable. 0 Channel done writeback disabled. 1 Channel done writeback enabled. Upon completion of descriptor processing, if the NT bit is set for global, or if the DN (done notification) bit is set in the header word of the descriptor, then notify the host by writing back the descriptor header with the writeback information shown in <a href="#">Table 12-51</a> . This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed.
60	AWSE	Always writeback status enable. 0 No special action. 1 At the completion of processing each descriptor, write back the status of primary and secondary EUs into the header dword. In this case, IWSE has no effect.

**Table 12-50. Crypto-Channel Configuration Register (CCCR) Field Descriptions (continued)**

Bits	Name	Description
61	NT	Notification type. This bit controls when the channel will generate channel done notification. Channel done notification can take the form of an interrupt or modified header writeback or both, depending on the state of the CDIE and CDWE control bits. 0 Global notification: The channel will generate channel done notification (if enabled) at the end of each descriptor. 1 Selected notification: The channel will generate channel done notification (if enabled) at the end of every descriptor with the DONE bit set in the descriptor header.
62	CDIE	Channel done interrupt enable. 0 Channel done interrupt disabled 1 Channel done interrupt enabled. Upon completion of descriptor processing, if the NT bit is set for global, or if the DN (done notification) bit is set in the header word of the descriptor, then notify the host by asserting an interrupt. Refer to <a href="#">Section 12.5.2, “Channel Interrupts,”</a> for complete description of channel interrupt operation.
63	—	Reserved. Set to zero.

Figure 12-73 shows the format of the header dword when the channel is configured to perform done notification via header writeback. A detailed description of the header dword fields used in writeback notification is provided in [Table 12-51](#).



**Figure 12-73. Header Dword Writeback Format**

**Table 12-51. Header Dword Writeback Field Descriptions**

Bits	Name	Description
0–7	DONE	When done writeback is used, then at the completion of descriptor processing this byte is written with the value 0xFF. To determine when done writeback is used, see the CDWE, NT, and CDIE fields in the channel configuration register (see <a href="#">Table 12-51</a> ).
8–34	—	Reserved.
35–36	ICCR0	Integrity check comparison result from primary. These bits are supplied by the primary EU when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
37–42	—	Reserved.
43–44	ICCR1	Integrity check comparison result from secondary. These bits are supplied by the secondary EU (if any) when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
45–63	—	Reserved.



### 12.5.1.2 Crypto-Channel Pointer Status Registers 1–4 (CCPSR<sub>n</sub>)

These registers contains status fields and counters which provide the user with status information regarding the channel's actual processing of a given descriptor.

Address	Channel_1 0x01110	Access: Read-only
	Channel_2 0x01210	
	Channel_3 0x01310	
	Channel_4 0x01410	

	0	2	3	7	8	11	12	15	16	19	20	23	24	31	
R	—		FF_COUNTER			—		G_STATE			—		S_STATE	CHN_STATE	
W															
Reset	All zeros														

	32	37	38	39	40	41	42	43	44	45	46	47	48	59	60	63
R	—		MI	MO	PR	SR	PG	SG	PRD	SRD	PD	SD	Error		PTR_DW	
W																
Reset	All zeros															

**Figure 12-74. Crypto-Channel Pointer Status Register**

Table 12-52 describes the channel pointer status register fields.

**Table 12-52. Crypto-Channel Pointer Status Register Field Descriptions**

Bits	Name	Description
0–2	—	Reserved.
3–7	FF_COUNTER	Fetch FIFO counter. This 5 bit counter indicates how many fetch pointers are currently stored in the FIFO.
8–10	—	Reserved.
12–15	G_STATE	Gather state machine state. This field reflects the state of the channel gather control state machine. The value of this field indicates which stage the channel is while performing gather function. <a href="#">Table 12-53</a> shows the meaning of all possible values of the G_STATE field. <b>Note:</b> G_State is documented for information only. The user will not typically care about the gather state machine.
16–19	—	Reserved.
20–23	S_STATE	Scatter state machine state. This field reflects the state of the channel scatter control state machine. The value of this field indicates which stage the channel is while performing scatter function. <a href="#">Table 12-53</a> shows the meaning of all possible values of the S_STATE field. <b>Note:</b> S_State is documented for information only. The user will not typically care about the scatter state machine.
24–31	CHN_STATE	State. State of the channel state machine. This field reflects the state of the channel control state machine. The value of this field indicates exactly which stage the channel is in the sequence of fetching and processing data descriptors. <a href="#">Table 12-54</a> shows the meaning of all possible values of the STATE field. <b>Note:</b> CHN_State is documented for information only. The user will not typically care about the channel state machine.
32–37	—	Reserved, set to zero

**Table 12-52. Crypto-Channel Pointer Status Register Field Descriptions (continued)**

Bits	Name	Description
38	MI	Multi_EU_IN. The Multi_EU_IN bit indicates whether data input snooping will be performed, as determined by the snoop type bit in the descriptor header. 0 Data input snooping by secondary EU disabled. 1 Data input snooping by secondary EU enabled.
39	MO	Multi_EU_OUT. The Multi_EU_OUT bit indicates whether data output snooping will be performed, as determined by the snoop type bit in the descriptor header. 0 Data output snooping by secondary EU disabled. 1 Data output snooping by secondary EU enabled.
40	PR	PRI_REQ. Request primary EU assignment. 0 Primary EU assignment request is inactive. 1 The channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the descriptor header register as long as this bit remains set. The PRI_REQ bit is set when descriptor processing is initiated by the channel and the Op_0 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the PRI_GRANT bit.
41	SR	SEC_REQ. Request secondary EU assignment. 0 Secondary EU assignment request is inactive. 1 The channel is requesting assignment of secondary EU to the channel. The channel will assert the EU request signal indicated by the Op_1 field in the descriptor header register as long as this bit remains set. The SEC_REQ bit is set when descriptor processing is initiated by the channel and the Op_1 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the SEC_GRANT bit.
42	PG	Primary EU granted. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
43	SG	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
46	PD	Primary EU done. The PRI_DONE bit reflects the state of the done interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.

**Table 12-52. Crypto-Channel Pointer Status Register Field Descriptions (continued)**

Bits	Name	Description
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the done interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
48–59	Error	Channel error status. This field reflects the error status of the channel. When a channel error interrupt is generated, this field will reflect the source of the error. The bits in the ERROR field are registered at specific stages in the descriptor processing flow. Once registered, an error can only be cleared only by resetting the channel or writing the appropriate registers to initiate the processing of a new descriptor. <a href="#">Table 12-55</a> lists the conditions which can cause a channel error and how they are represented in the ERROR field.
60–63	PTR_DW	PTR_DW indicates which pointer-dword is currently being processed by the channel. <a href="#">Table 12-56</a> shows the meaning of all possible values of the PTR_DW field.

[Table 12-53](#) shows the values for G\_STATE and S\_STATE fields, which are the states for the gather and scatter state machines.

**Table 12-53. G\_STATE and S\_STATE Field Values**

Value	Gather State Machine
0x0	GS_IDLE
0x1	GS_LOAD_POINTER
0x2	GS_LOAD_POINTER_DONE
0x3	GS_LOAD_NEXT_POINTER
0x4	GS_PROCESS_POINTER
0x5	GS_TRANS_BLOCK
0x6	GS_TRANS_BLOCK_DONE
0x7	GS_TRANS_BYTES
0x8	GS_TRANS_BYTES_DONE
0x9	GS_INC_PAIR_PTR
0xA	GS_UPDATE
0xB	GS_DONE
0xC	GS_ERROR
0xD	GS_RELOAD
0xE	GS_TRANS_INBOUND
0xF	GS_TRANS_INBOUND_DONE

Table 12-54 shows the values of crypto-channel states.

**Table 12-54. CHN\_STATE Field Values**

Value	Channel State	Value	Channel State
0x00	IDLE	0x20	DELAY_SEC_DONE
0x01	PROCESS_HEADER	0x21	TRANS_REQUEST_READ
0x02	FETCH_DESCRIPTOR	0x22	EVALUATE_RESET
0x03	CHANNEL_DONE	0x23	RESET_WRITE_RESET_PRI
0x04	CHANNEL_DONE_IRQ	0x24	RESET_RELEASE_PRI_CHA
0x05	CHANNEL_DONE_WRITEBACK	0x25	RESET_WRITE_RESET_SEC
0x06	CHANNEL_DONE_NOTIFICATION	0x26	RESET_RELEASE_SEC_CHA
0x07	CHANNEL_ERROR	0x27	RESET_CHANNEL
0x08	REQUEST_PRI_CHA	0x28	WRITE_DATASIZE_PRI_POST
0x09	INC_DATA_PAIR_POINTER	0x29	RESET_RELEASE_ALL
0x0A	DELAY_DATA_PAIR_UPDATE	0x2A	RESET_RELEASE_ALL_DELAY
0x0B	EVALUATE_DATA_PAIRS	0x2B	REQUEST_SEC_CHA
0x0C	WRITE_RESET_PRI	0x2C	WRITE_DATASIZE_SEC
0x0D	RELEASE_PRI_CHA	0x2D	WRITE_ICV_SIZE
0x0E	WRITE_RESET_SEC	0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x0F	RELEASE_SEC_CHA	0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x10	PROCESS_DATA_PAIRS	0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x11	WRITE_MODE_PRI	0x31	DELAY_1CYCLE
0x12	WRITE_MODE_SEC	0x33	TRANS_EXTENT_READ
0x13	WRITE_DATASIZE_PRI	0x34	TRANS_EXTENT3
0x14	DELAY_RNGA_DONE	0x35	TRANS_EXTENT4
0x15	WRITE_DATASIZE_SEC_SNOOPIN	0x36	XOR_WRITE_READ_REG
0x16	TRANS_REQUEST_WRITE_SNOOPIN	0x37	DELAY_SEC_DONE_TLS
0x17	DELAY_PRI_SEC_DONE	0x38	MAC_TO_CIPHER
0x18	TRANS_REQUEST_WRITE	0x39	MAC_TO_CIPHER_DONE
0x19	WRITE_KEY_SIZE	0x3A	READ_PRI_STATUS
0x1B	DELAY_PRI_DONE	0x3C	READ_SEC_STATUS
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT	others	Reserved
0x1F	TRANS_REQUEST_READ_SNOOPOUT		

Table 12-55 shows the bit positions of each potential error. Multiple errors are possible.

**Table 12-55. Crypto-Channel Pointer Status Register Error Field Descriptions**

Value	Error
48	DOF. Double fetch FIFO write overflow error. This bit is set when the channel fetch FIFO is full, SOF is set, and another write has been made to the fetch FIFO. When this bit is set the channel will stop, and an error interrupt will be activated. The channel will not start again until a continue or reset is given via the CCR register. This bit can be cleared by writing '1' to this bit in the CPSR Register.
49	SOF. Single fetch FIFO write overflow Error. This bit is set when the channel fetch FIFO is full and another write has been made to the fetch FIFO. The channel will set this bit and activate an error interrupt. The channel continues processing, but the descriptor pointer is lost. The host must clear this bit by writing '1' to this bit in the CPSR register.
50	MDTE. A master data transfer error was received from the master bus interface. When the SEC, while acting as a bus master, detects an error, the controller passes the error to the channel in use. The channel halts and activates an interrupt. The channel can only be restarted by writing a '1' to the continue or reset bit in the channel configuration register, or resetting the whole SEC.

**Table 12-55. Crypto-Channel Pointer Status Register Error Field Descriptions (continued)**

Value	Error
51	Scatter/gather data length zero error. A zero length scatter/gather data pointer was detected.
52	Fetch pointer zero error. An all zero fetch pointer was detected.
53	Illegal descriptor header. Possible causes of an illegal descriptor header are: <ul style="list-style-type: none"> <li>• Invalid primary EU indicated by op0 field in descriptor header.</li> <li>• Invalid secondary EU indicated by op1 field in descriptor header.</li> <li>• Descriptor type field in descriptor headers indicates secondary EU transaction when not in snoop mode</li> </ul>
54	Invalid EU assignment request. Indicates the channel was assigned one or more EUs not requested by the descriptor header.
55	EU error detected. An EU assigned to this channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register.
56	Gather boundary error. Indicates a gather pointer straddles both a primary and secondary EU's data transfer.
57	Gather return/length error. Indicates the total data size covered by a gather link table did not match the total data size from the main descriptor.
58	Scatter boundary error. Indicates a scatter pointer straddles both a primary and secondary EU's data transfer.
59	Scatter return/length error. Indicates the total data size covered by a scatter link table did not match the total data size from the main descriptor.

**NOTE**

The EU error bit (bit 55) can only be cleared by first clearing the error source in the assigned EU which caused it to be set.

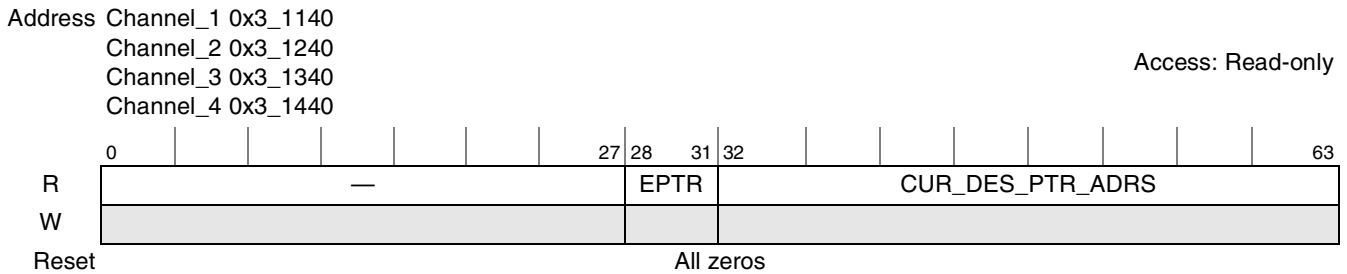
Table 12-56 shows the possible values of the PTR\_DW field in the CPSR.

**Table 12-56. Channel Pointer Status Register PTR\_DW Field Values**

Value	Error
0x00	Processing header or pointer dword 0
0x01	Processing pointer dword 1
0x02	Processing pointer dword 2
0x03	Processing pointer dword 3
0x04	Processing pointer dword 4
0x05	Processing pointer dword 5
0x06	Processing pointer dword 6
0x07	Complete (or not yet begun) processing of header dword and pointer dwords
0x08–FF	Reserved

### 12.5.1.3 Crypto-Channel Current Descriptor Pointer Registers 1–4 (CCDPR<sub>n</sub>)

The CCDPRs, shown in [Figure 12-75](#), contain the address of the descriptor that the channel is currently processing.



**Figure 12-75. Crypto-Channel Current Descriptor Pointer Register**

The bits in the current descriptor pointer register perform the functions described in [Table 12-57](#).

**Table 12-57. Channel Current Descriptor Pointer Register Field Descriptions**

Bits	Name	Description
0–27	—	Reserved. Set to zero.
28–31	EPTR	Extended pointer. Concatenated as the top 4 bits of the CUR_DES_PTR_ADRS when EAE is high (see the EAE bit in Table 1-51 on page 1-116).
32–63	CUR_DES_PTR_ADRS	Current descriptor pointer address. Pointer to system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the channel requests a fetch of a descriptor from the controller. The value from the fetch FIFO is transferred to the current descriptor pointer register immediately after the fetch is completed. This address will be used as the destination for writeback of the modified header dword, if header writeback notification is enabled.

### 12.5.1.4 Fetch FIFO Address Registers 1–4 (FFn)

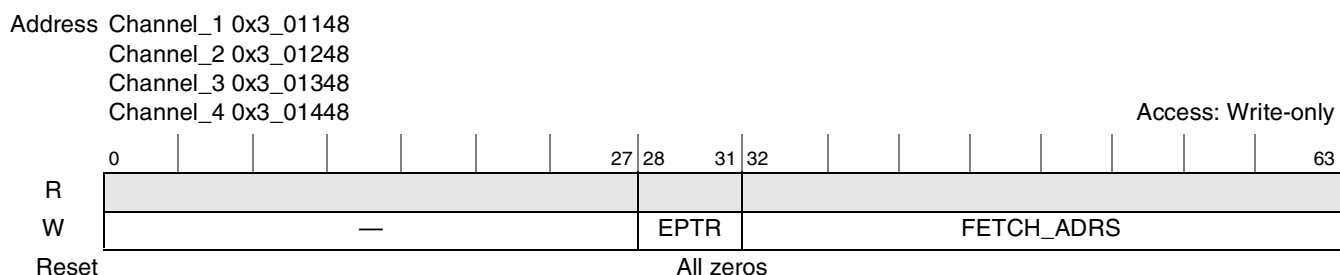
Each channel contains a fetch address FIFO to store a queue of pointers to descriptors for the channel to process.

The fetch address FIFOs, shown in [Figure 12-76](#), contain the addresses of the first byte of descriptors to be processed. In typical operation, the host CPU creates a descriptor in memory containing all relevant mode and location information for the SEC, then launches the SEC by writing the address of the descriptor to the fetch FIFO.

The fetch FIFO can hold up to 24 descriptor pointers at a time. When the end of the current descriptor is reached, the descriptor pointed to by the next location in the fetch FIFO is read to launch the next descriptor.

The fetch address is written into the FIFO only if the write includes the least-significant byte (bits 56–63). If extend address enable is high (see the EAE bit in [Table 12-50](#)), then the extended fetch address must be written before the fetch address or concurrent with it. Specifying a FETCH\_ADRS of 0 causes the channel to generate an error and stop.

Writing a descriptor pointer to the fetch FIFO while the FIFO is full will result in a single overflow interrupt to advise the user that the descriptor pointer was not successfully written to the fetch FIFO. The channel will continue processing and software can check the fetch FIFO counter in the crypto-channel pointer status register before attempting to re-enqueue the descriptor pointer. If a second descriptor pointer is written to the fetch FIFO before the single overflow error is cleared, the channel generates a double overflow error interrupt and stop processing descriptors. The channel can be restarted by setting the continue bit in the crypto-channel configuration register, or completely reset by writing the reset bit in the same register.



**Figure 12-76. Fetch FIFO**

[Table 12-58](#) describes the fetch FIFO fields.

**Table 12-58. Fetch FIFO Field Descriptions**

Bits	Name	Description
0–27	—	Reserved — Set to zero.
28–31	EPTR	Extended pointer. Concatenated as the top 4 bits of the FETCH_ADRS when EAE is high (see the EAE bit in <a href="#">Table 1-51</a> on page 1-116).
32–63	FETCH_ADRS	Fetch address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

### 12.5.1.5 Crypto-Channel 1–4 Descriptor Buffers [0–7] (DB<sub>n</sub>[0–7])

The descriptor buffers (DBs) consists of 8 dword registers (DB<sub>n</sub>[0–7]), and contain the current descriptor being processed by the channel. These registers are read-only, since the descriptor is always fetched from system memory.

The write of any valid pointer to the fetch FIFO will cause the channel to read 64 contiguous bytes beginning at the fetch address into the descriptor buffer.

For more information about the fields in a descriptor, see [Section 12.3.1, “Descriptor Structure.”](#)

	0	15	16	17	23	24	31	32	63
DB0	Header							Reserved	
DB1	Length0	J1	Extent0	—				Pointer0	
DB2	Length1	J2	Extent1	—				Pointer1	
DB3	Length2	J3	Extent2	—				Pointer2	
DB4	Length3	J4	Extent3	—				Pointer3	
DB5	Length4	J5	Extent4	—				Pointer4	
DB6	Length5	J6	Extent5	—				Pointer5	
DB7	Length6	J7	Extent6	—				Pointer6	
Address	Channel_1 0x3_1180–0x3_11BF, Channel_2 0x3_1280–0x3_12BF, Channel_3 0x3_1380–0x3_13BF, Channel_4 0x3_1480–0x3_14BF								

Figure 12-77. Descriptor Buffer Format

### 12.5.1.6 Link Table Buffer Registers (Scatter or Gather)—LTB0–3

The link table buffer consists of 4 dword registers (LTB0–LTB3), and contains the link table (scatter or gather) being processed by the channel, when scatter/gather is in use. These registers are read-only, since the link table is always fetched from system memory.

Any descriptor length/pointer dword with the jump bit set will cause the channel to read 32 contiguous bytes beginning at the pointer address into the link table buffer. Additionally, any link table entry with the next bit set causes the channel to read 32 contiguous bytes into the link table buffer. As the data may not be so scattered as to require 4 link table entries to gather it, automatically reading 32 bytes (4 link table entries) is pre-fetching, and is done for performance reasons. Proper use of the return and next bits is required to avoid problems arising from pre-fetching.

For more information about the fields in a link table, see [Section 12.3.4, “Link Table Format.”](#)



0		15	16	21	22	23	24	27	28	31	32	63
LTB0	SEGLN	—	R	N	—	EPTR	SEGPTR					
LTB1	SEGLN	—	R	N	—	EPTR	SEGPTR					
LTB2	SEGLN	—	R	N	—	EPTR	SEGPTR					
LTB3	SEGLN	—	R	N	—	EPTR	SEGPTR					

Figure 12-78. Link Table Buffer

## 12.5.2 Channel Interrupts

The channel can assert both DONE and ERROR interrupts to the controller. When the interrupt generation conditions have been met, the channel asserts the appropriate interrupt. The status of the registered channel interrupts is available in the controller interrupt status register. The channel does not have an internal interrupt controller, but the SEC controller can be programmed to block channel interrupts via its interrupt mask register (see [Section 12.6.5.2, “Interrupt Mask Register \(IMR\)”](#)).

### 12.5.2.1 Channel Done Interrupt

Whether and when a channel DONE interrupt is generated depends on the setting of the channel configuration register NT and CDIE bits in the CCR (see [Figure 12-72](#)). Assuming the CDIE (channel done interrupt enable) is set, the channel will generate an interrupt event after every successfully completed descriptor (notification type set to global), or after each successfully completed descriptor with the DN (done notification) bit set in the header word of the descriptor.

Even if multiple channel done interrupt events are generated by a channel before the first can be cleared by the host, the interrupt events are not lost. The controller queues channel done interrupts from each channel (see [Section 12.6.4, “Controller Interrupts”](#)).

### 12.5.2.2 Channel Error Interrupt

The channel error interrupt is generated when an error condition occurs during descriptor processing. The channel error interrupt will be asserted as soon as the error condition is detected. The type of error condition is reflected the ERROR field of the channel pointer status register (CPSR). Refer to [Table 12-55](#) for a complete listing of error types.

### 12.5.2.3 Channel Reset

Channel reset is asserted when the host sets the RESET bit in the channel configuration register (CCR). The effect of software reset on the channel varies according to what the channel is doing when the bit is set:

- If the RESET bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request by asserting the release output signals. The channel then resets all the registers, clears the RESET bit and returns the control state machine to the idle state.
- If the RESET bit is set after the channel has been dynamically assigned an EU, the channel requests a write from the controller to set the software reset bit of the EU. A write to reset the secondary

(MDEU) EU is also requested if one has been reserved for snooping. The channel then asserts the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the RESET bit and returns the control state machine to the idle state.

## 12.6 Security Controller

The controller within the SEC is responsible for overseeing the operations of the execution units (EUs), the interface to the host processor, and the management of the channels. The controller interfaces to the host via the master/slave bus interface and to the channels and EUs via internal buses. All transfers between the host and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Provide arbitration for bus access and control bus accesses
- Control the internal bus accesses to the EUs
- Provide arbitration for channels requesting EUs and assign EUs to channels
- Monitor interrupts from channels and pass to host
- Realign read and write data to the proper byte alignment

### 12.6.1 Assignment of EUs to Channels

Assignment of an EU to a channel is done dynamically. The channel requests an EU, the controller checks to see if the requested EU is available, and if it is, the controller grants the channel assignment of the EU.

If an EU is available for a channel when requested, the controller will assert the grant signal pertaining to the request from the channel. The grant signal will remain asserted until the channel is done and releases the EU.

In some cases, a channel may request two EUs. The channel will do this by first requesting the primary EU, then requesting the secondary EU. Once the controller has granted both EUs, this channel is then capable of requesting that the secondary EU snoop the bus. Snooping status is indicated in the MI and MO bits of [Table 12-52](#).

In all cases, the controller assigns the primary EU to a requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing any grants to a channel which is requesting two EUs.

Since there are multiple channels in the SEC, they must arbitrate for access to execution units. To accomplish this, the controller implements one arbiter for each EU and one arbiter for the internal system bus. These are snapshot arbiters, which means they operate as follows:

- When there are requests, the arbiter records the set of requests waiting (that is, it takes a snapshot)
- Then it satisfies those requests as the resource becomes available.
- When all requests in the snapshot have been satisfied, the arbiter takes another snapshot.

To choose which request to grant within a given snapshot, the arbiters can use either a weighted priority-based or round-robin scheme, depending on the values of CHN3\_EU\_PR\_CNT and CHN4\_EU\_PR\_CNT in the master control register (see [Section 12.6.5.7, “Master Control Register](#)

(MCR”). If both CHN3\_EU\_PR\_CNT and CHN4\_EU\_PR\_CNT are set to non-zero values, the arbiter will implement the weighted priority scheme. If both are zero, the arbitration will be round-robin. Setting only one of the CHN\_EU\_PR\_CNT fields to a non-zero value will result in unpredictable operation.

### 12.6.1.1 Channel Priority Arbitration

When arbitrating on the priority scheme, the priority is as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHN3\_EU\_PR\_CNT or CHN4\_EU\_PR\_CNT expired
- Channel 3—Third priority, unless CHN4\_EU\_PR\_CNT expired
- Channel 4—Lowest priority, until CHN4\_EU\_PR\_CNT expired

Initially, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, the CHN3\_EU\_PR\_CNT and CHN4\_EU\_PR\_CNT fields are implemented in the master control register (MCR). The value of these fields determines how many times channel 3 or channel 4 can be refused access to an EU in favor of a higher priority channel. A counter is implemented in the arbiter for each of these entities. When the channel has lost arbitration the number of times specified in its CHN\_EU\_PR\_CNT field, then that channel has the 2nd highest priority when the requested EU becomes available. CHN1 always has the highest priority, but it cannot make back to back requests, so the 2nd highest priority channel will be serviced upon completion of the current CHN1 operation.

It is permissible for the CHN\_EU\_PR\_CNT values to be different from the CHN\_BUS\_PR\_CNT values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice-versa.

### 12.6.1.2 Channel Round-Robin Arbitration

In round-robin arbitration, requesting channels are granted access in rotating numerical order: 1, 2, 3, 4, 1, 2, ... etc.

## 12.6.2 Bus Transfers

The controller in the SEC has the ability to be a bus master or a slave. This means that the controller can issue read and write commands to the bus, and it can also be written to and read from by the host.

The controller is the sole bus master within the SEC. All other modules are slave-only devices. A channel may request access to system resources including the bus. In these cases, the channel must provide the starting address of the transfer for the bus(es) requested. All subsequent addresses are generated by the controller. All addresses are sequential.

The controller is involved in transfers on the internal system bus and the internal bus. For channel-controlled access, the channels make requests to the controller to perform data transfers. The channel specifies data lengths and addresses for the internal and system buses. Multiple channels may request use of the controller at the same time, so the controller performs arbitration to choose a channel. The controller then services the request and performs the required transfer. Most transfers involve not only

the internal bus, but also the system bus with the controller as bus master. Here are examples of the various types of transfers:

1. Obtaining a descriptor:
  - Channel makes request, arbitrates for attention from controller
  - Controller arbitrates for use of the system bus and performs read from external memory
  - Controller sends descriptor to channel over the internal bus
2. Transferring data length parameter from channel to EU
  - Channel makes request, arbitrates for attention from controller
  - Controller transfers data from channel to EU over the internal bus
3. Obtaining input data from external memory for input to an EU
  - Channel makes request, arbitrates for attention from controller
  - Controller arbitrates for use of the system bus and performs read from external memory
  - Controller sends data to EU over the internal bus. If in-snooping, data is sent to two EUs
4. Writing output data from an EU back to external memory
  - Channel makes request, arbitrates for attention from controller
  - Controller begins reading data from the EU into a controller FIFO. If out-snooping, the same data is also read by another EU. Meanwhile, the controller arbitrates for use of the internal system bus.
  - Controller performs write to external memory

### 12.6.2.1 Arbitration for Use of the Controller and Buses

The controller attempts to maximize use of the system bus by grouping outstanding bus requests from the channels by request type (read or write). The controller performs all write requests to the internal system bus, followed by all read requests, then repeat.

Within a request type, the controller grants bus access via the same snapshot scheme that is used for granting EUs. If, for example, the controller is performing writes, it takes a snapshot of the current write requests, satisfies them as the bus becomes available, then takes another snapshot of write requests, and repeats. If there are no more write requests in a snapshot, the arbiter switches to handling reads. It repeatedly takes snapshots of the reads waiting and satisfies them until there are no more read requests in a snapshot. It then switches back to handling writes, etc.

As with arbitration for EUs, controls for setting channel priorities are in the master control register (see [Section 12.6.5.7, “Master Control Register \(MCR\)”](#)), and the same two methods are available for selecting which request to satisfy within a snapshot. If both `CHN3_BUS_PR_CNT` and `CHN4_BUS_PR_CNT` are set to non-zero values, the arbiter will implement the weighted priority scheme (see [Section 12.6.1.1, “Channel Priority Arbitration”](#)). If both are zero, the arbitration will be round-robin (see [Section 12.6.1.2, “Channel Round-Robin Arbitration”](#)). Setting only one of the `CHNx_BUS_PR_CNT` fields to a non-zero value results in unpredictable operation. When the buses are granted to a channel, they are granted until the channel transfer is completely satisfied.

The SEC does not dynamically adjust its own transaction priorities. System software, however, can adjust SEC transaction priority in real-time, with the change in priority taking effect immediately.

### 12.6.2.2 System Bus Master Reads

This section contains more detail on the sequence of events for a system bus read with the controller as master:

1. Channel asserts its bus read request to the controller.
2. Channel furnishes external read address, internal write address, and transfer length.
3. Controller sends request acknowledge to channel.
4. Controller asserts request to the system bus.
5. Controller waits for system bus read to begin.
6. When bus read begins, controller receives data from the master interface and performs a write to the appropriate internal address supplied by the channel. Data may be realigned byte-wise by the controller if either:
  - the read did not begin on a 32-bit word boundary, or
  - the previous write to an execution unit's input FIFO did not end on a 32-bit word boundary.
7. Transfer continues until the bus read is completed and the controller has written all data to the appropriate internal address. The master interface continues making bus requests until the full data length has been read.

When the SEC performs a transaction as master, it is possible for the intended slave to terminate the transfer due to an error. The SEC's transaction requests are posted to the MPC8533E target queue, after which the MPC8533E takes responsibility for completing the transaction or signaling error. An error in an SEC initiated transaction is also reported by the SEC via the channel interrupt status register. The host will be able to determine which channel generated the interrupt by checking the ISR for the channel ERROR bit.

### 12.6.2.3 System Bus Master Writes

This section contains more detail on the sequence of events for an system bus write with the controller as master:

1. Channel asserts its bus write request to the controller.
2. Channel furnishes internal read address, external write address, and transfer length.
3. Controller sends request acknowledge to channel.
4. Controller performs a read from the appropriate internal address supplied by the channel, loads the write data into its FIFO, asserts a request to the system bus and waits for the system bus to become available.
5. When the system bus becomes available, controller writes data from its FIFO to the master interface.

While the controller is satisfying a requests from one channel, it can still acknowledge and queue requests from the other channels.

### 12.6.2.4 System Bus Slave Transactions (Reads and Writes)

The controller also acts as a bus slave. As a slave, the controller simply responds to read and write commands from the bus. When a write command is received from the bus, the controller takes the data

from the slave interface and sends it to whichever internal location is indicated by the address. For a read, the controller goes to the internal location and fetches the requested data from the specified address. The SEC internal memory space must be accessed on modulo-4 boundaries to avoid invalid data or unpredictable operation.

### 12.6.3 Snooping by Caches

All SEC transactions are snooped by the e500 coherency module (ECM). This is part of the wiring of the SEC interface and requires no user intervention.

### 12.6.4 Controller Interrupts

All interrupt outputs from the channels, EUs, and bus interface are fed to the controller as interrupt conditions. In addition, the controller itself detects some interrupt conditions. The controller maintains an interrupt status register (ISR) with bits corresponding to all of these possible interrupt conditions. If an interrupt condition occurs and the corresponding bit of the interrupt mask register (IMR) is high, then the associated interrupt status register bit is set, indicating the presence of a pending interrupt. Whenever any bits are set in the interrupt status register, the controller asserts its internal interrupt signal to the host.

To handle an interrupt, the host must read the interrupt status register (ISR) to determine the source. It may then need to do further reads of interrupt status registers of other blocks to get more detailed information about the cause. In some cases, the host may need to take action to clear the root cause of the interrupt.

After that, the host can clear the desired bit of the interrupt status register by writing a 1 to the corresponding bit of the interrupt clear register (ICR) (and then write zeros to the end of interrupt register in the PIC to enable additional SEC interrupts). If the cause of the interrupt condition has not been cleared, or if there is some other interrupt condition from the same source, then the interrupt status register bit will clear for a cycle and go high again, and the interrupt output line to the host remains high. If the interrupt status register bit is successfully cleared and no other interrupt conditions are present, the controller negates its interrupt output. If any interrupts are still pending in the interrupt status register, the interrupt output remains asserted.

Note that EU interrupt conditions may be blocked at two different levels. There is an interrupt control register in each EU which can block particular interrupt conditions before they reach the EU's interrupt status register, and in addition, bits of the controller's interrupt mask register must be set to allow interrupt conditions to reach the interrupt status register. Interrupt conditions from the channels and controller can only be blocked through the interrupt mask register.

For typical operation it is suggested that the interrupt mask register be programmed as follows:

- Leave channel interrupts enabled, while masking interrupts from the EUs.

Errors or done signals coming from the EUs eventually cause the channel to signal an error or done interrupt.

A channel can generate frequent interrupts, especially if it is configured to interrupt at the completion of each descriptor. To make sure that the host receives the right number of interrupts, each channel done interrupt has a special queuing feature. If multiple channel done interrupts are generated before the first is cleared, then the additional interrupts are queued by the controller. When the host clears a channel

interrupt, if there are no other interrupts queued from that channel, then the channel done interrupt is negated. If other interrupts remain in the queue, the controller negates the interrupt for one cycle and then re-asserts it again. Interrupts are queued separately for each channel.

The SEC generates a single interrupt to the MPC8533E embedded programmable interrupt controller (PIC). See the Programmable Interrupt Controller (PIC) chapter for additional information on the PIC. The user allows interrupts from the SEC to be reported to the CPU by clearing the mask bit in the associated vector/priority register of the PIC.

## 12.6.5 Controller Registers

The controller registers are described in detail in the following sections.

### 12.6.5.1 EU Assignment Status Register (EUASR)

The EUASR, displayed in [Figure 12-79](#), is used to check the assignment status of an EU to a particular channel. When an EU is already assigned, it is inaccessible to any other channel.

Offset 0x3_1028								Access: Read																
	0	3	4	7	8	11	12	15	16	19	20	23	24	27	28	31								
R	—			AFEU			—			MDEU			—			AESU			—			DEU		
W																								
Reset	All ones			All zeros			All ones			All zeros			All ones			All zeros			All ones			All zeros		
Offset 0x0104 (OR0)																								
	32			43			44			47			48			63								
R	—			KEU			—			PKEU			—			RNG								
W																								
Reset	0x000F			All zeros			All ones			All zeros			All ones			All zeros								

**Figure 12-79. EU Assignment Status Register (EUASR)**

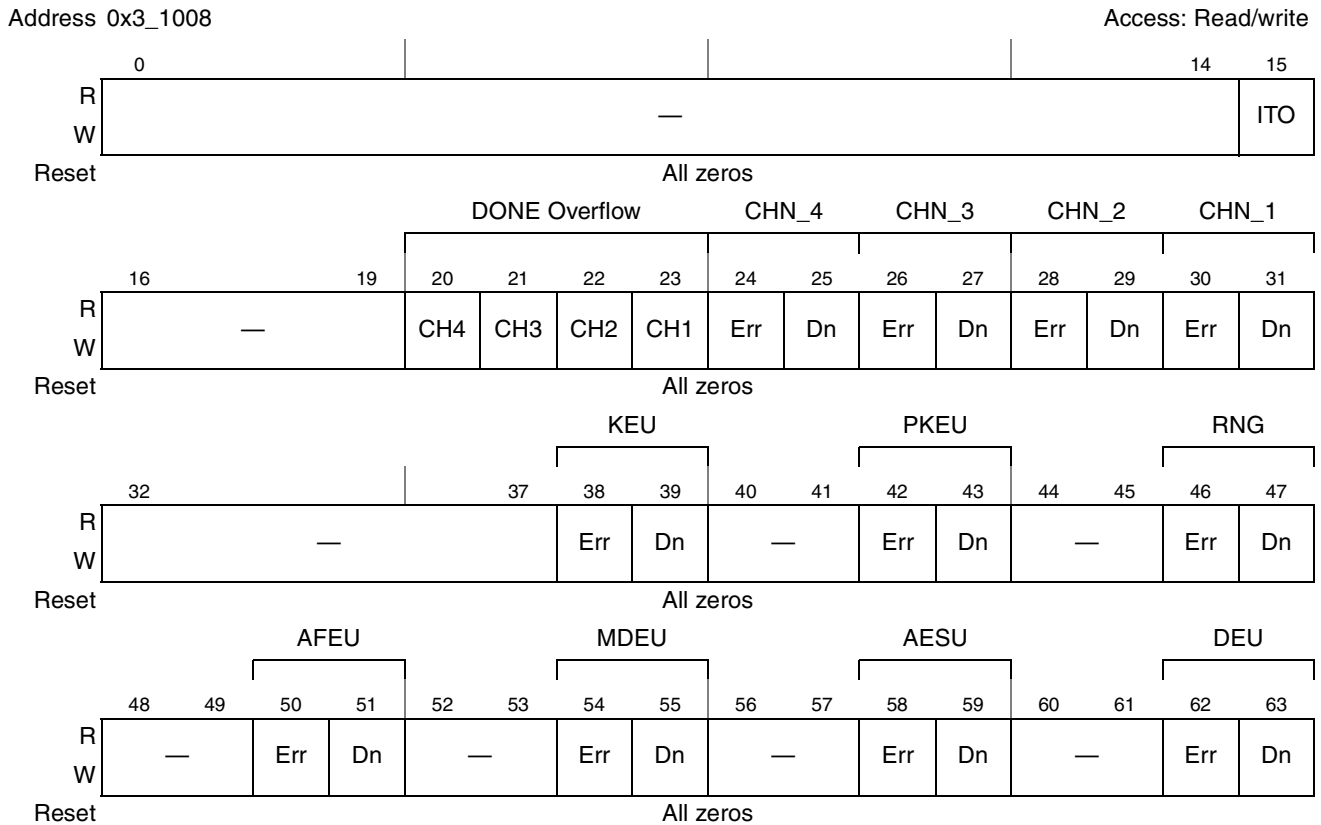
A four-bit field (see [Table 12-59](#)) indicates the channel to which an EU is assigned.

**Table 12-59. Channel Assignment Value**

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0xA–0xE	Undefined
0xF	Unavailable

### 12.6.5.2 Interrupt Mask Register (IMR)

The SEC controller generates the single interrupt output from all possible interrupt sources. These sources can be individually enabled by the interrupt mask register. If enabled, the interrupt source value, when active, is captured into the interrupt status register. Figure 12-80 shows the bit positions of each potential interrupt source. Each interrupt source is individually enabled by setting its corresponding bit. At reset, all bits are disabled. The bit fields are described in Table 12-60.



**Figure 12-80. Interrupt Mask Register (IMR)**

Table 12-60 describes the register field names in the interrupt mask register, interrupt status register, and interrupt clear register.

**Table 12-60. Field Names in Interrupt Mask, Interrupt Status, and Interrupt Clear Registers**

Bits	Name	Description
15	ITO	Internal time out. 0 No internal time out 1 An internal time out was detected The internal time out interrupt is triggered by the controller if a slave access to an SEC register does not result in successful data transfer within 16 clock cycles. With ITO enabled the SEC controller terminates the transaction and signals and interrupt.
20–23	Done Overflow	Done overflow. 0 No done overflow 1 Done overflow error. Indicates that more than 15 done interrupts were queued from the interrupting channel without an interrupt clear.



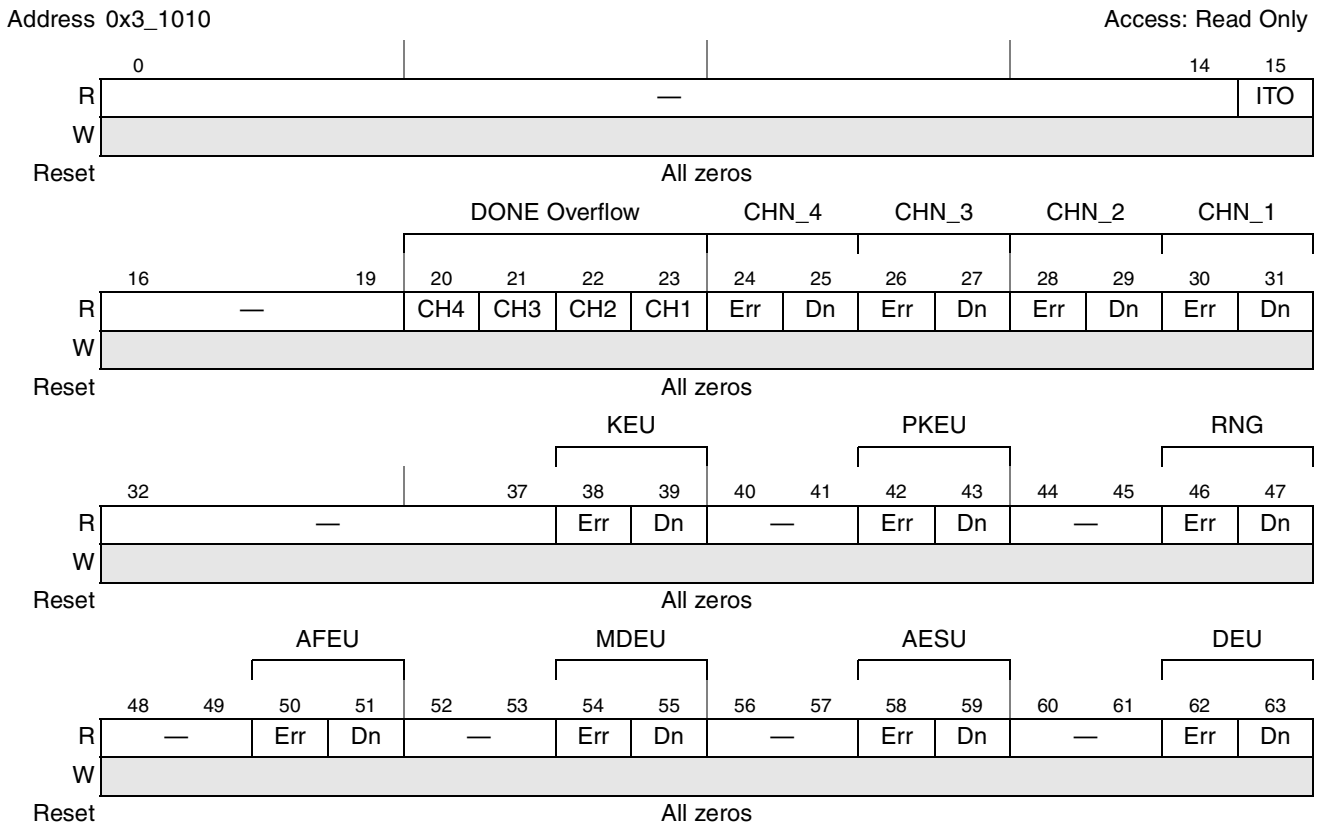
**Table 12-60. Field Names in Interrupt Mask, Interrupt Status, and Interrupt Clear Registers (continued)**

Bits	Name	Description
Multiple	CHN_Err_Dn	Each of the 4 channels has error and done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.
Multiple	EU_Err_Dn	Each of the execution units has error and done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.
Multiple	—	Reserved, set to zero.

### 12.6.5.3 Interrupt Status Register (ISR)

The ISR contains fields representing all possible sources of interrupts. The interrupt status register is cleared either by a reset, or by writing the appropriate bits active in the interrupt clear register.

Figure 12-81 shows the bit positions of each potential interrupt source.



**Figure 12-81. Interrupt Status Register (ISR)**

### 12.6.5.4 Interrupt Clear Register (ICR)

The interrupt clear register provides a means of clearing the interrupt status register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output signal  $\overline{IRQ}$  (assuming the cleared bit in the ISR is the only interrupt source). If the input source to the ISR is a steady-state signal that remains active, the appropriate ISR bit, and subsequently  $\overline{IRQ}$ , will be re-asserted shortly thereafter. Figure 12-82 shows the bit positions of each interrupt source that can be cleared by this register. The complete bit definitions for the ICR can be found in Figure 12-82.

When an ICR bit is written, it automatically clears itself one cycle later. That is, it is not necessary to write a '0' to a bit position which has been written with a '1.'

#### NOTE

Interrupts are registered and sent based upon the conditions that cause them. If the cause of an interrupt is not removed, the interrupt will return a few cycles after it has been cleared using the ICR.

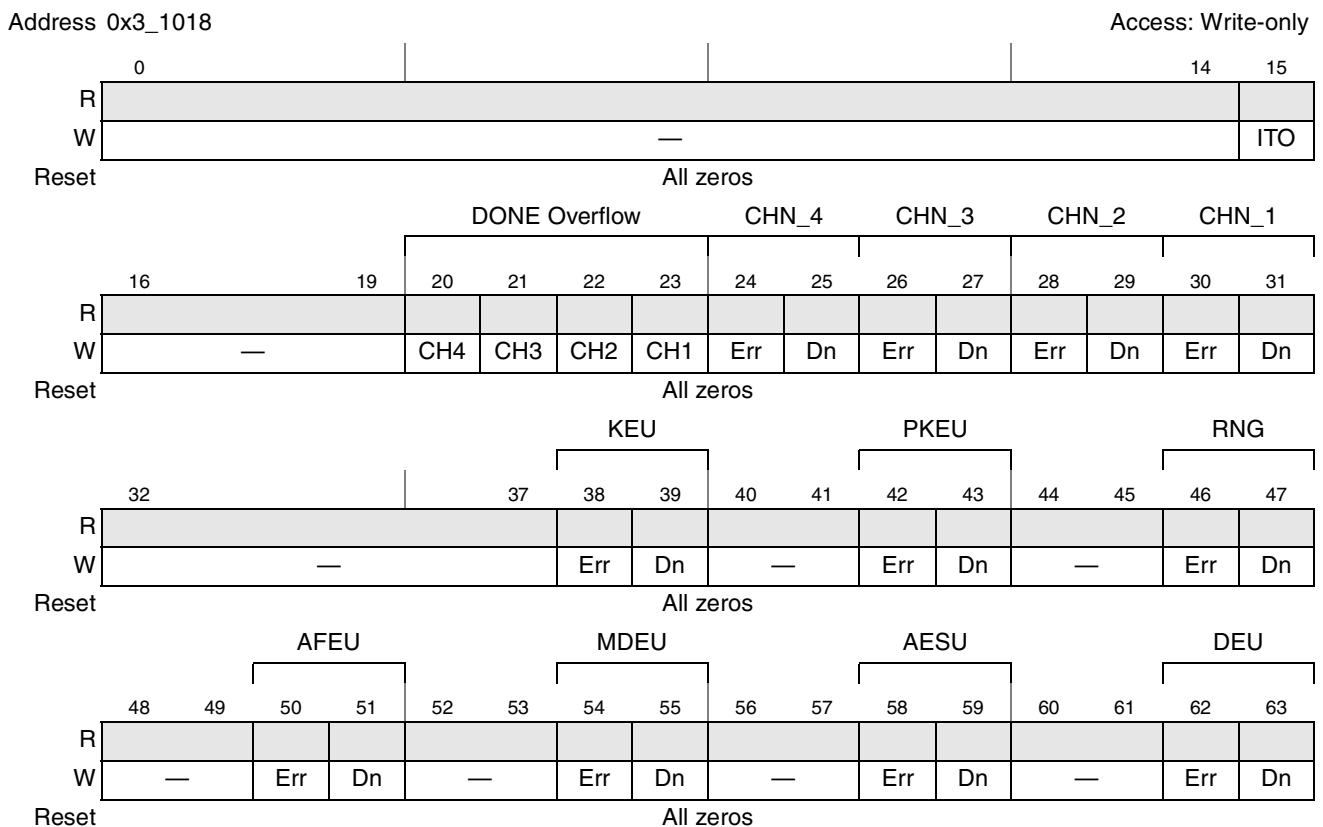
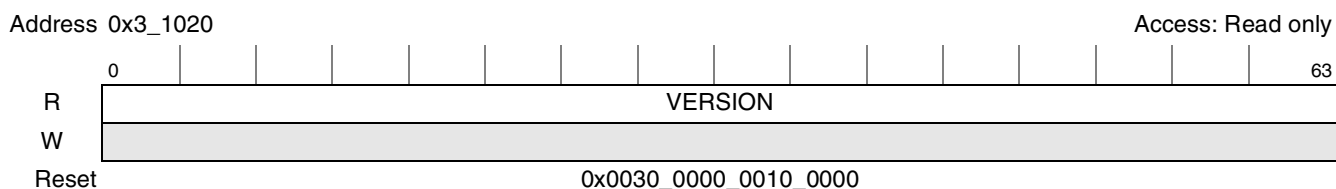


Figure 12-82. Interrupt Clear Register (ICR)

### 12.6.5.5 ID Register

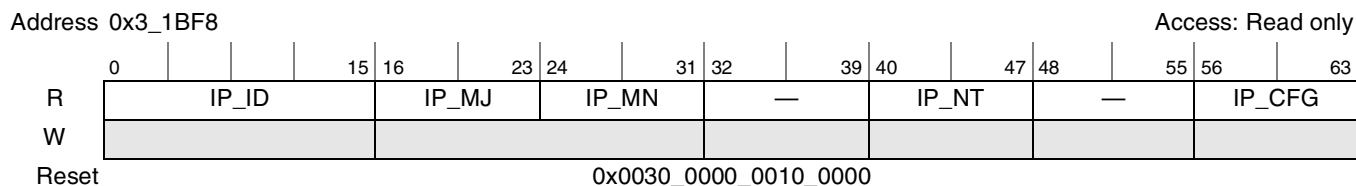
The read-only ID register, displayed in [Figure 12-83](#), contains a 64-bit value that uniquely identifies the version of the SEC 2.1. The value of this register is always 0x0030\_0000\_0010\_0000.



**Figure 12-83. ID Register (ID)**

### 12.6.5.6 IP Block Revision Register

The read-only IP block revision register, displayed in [Figure 12-84](#), contains a 64-bit value that encodes implementation and integration information specific of the SEC 2.1 as used in the MPC8533E. The value of this register is 0x0030\_0000\_0010\_0000. In general, the ID register is used by the device driver to recognize SEC capabilities, while the IP block revision register is used to track design revisions.



**Figure 12-84. IP Block Revision Register**

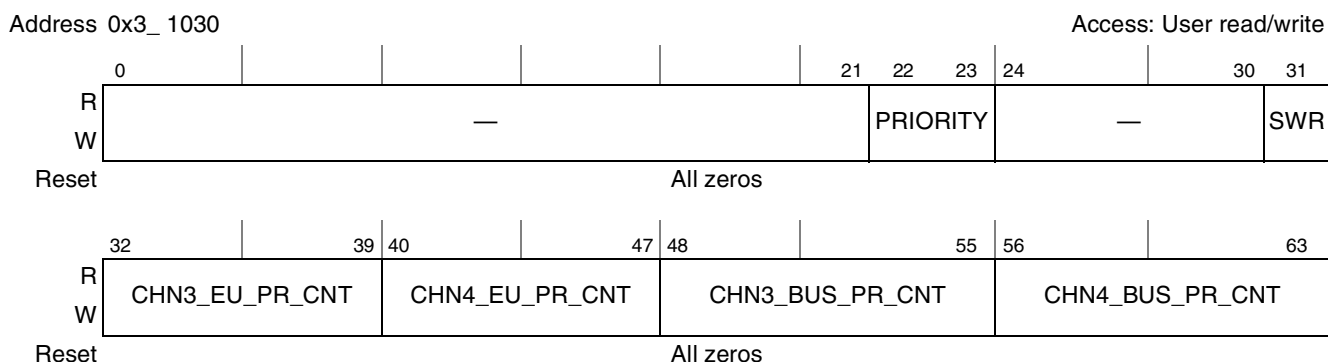
[Table 12-61](#) describes the fields of the IP block revision register.

**Table 12-61. Channel Current Descriptor Pointer Register Signals**

Bits	Name	Description
0–15	IP_ID	IP block identifier
16–23	IP_MJ	IP major revision number
24–31	IP_MN	IP minor revision number
32–39	—	Reserved
40–47	IP_INT	IP block integration options
48–55	—	Reserved
56–63	IP_CFG	IP block configuration options

## 12.6.5.7 Master Control Register (MCR)

The MCR, shown in [Figure 12-85](#), controls certain functions in the controller and provides a means for software to reset the SEC.



**Figure 12-85. Master Control Register (MCR)**

[Table 12-62](#) describes the master control register fields.

**Table 12-62. Master Control Register (MCR) Field Descriptions**

Bits	Name	Description
0–21	—	Reserved
22–23	Priority	Priority on master bus. The setting of these bits determines the transaction priority level the SEC asserts to the MPC8533E internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software may change the SEC priority level in real-time. 00 Lowest priority (default) 01 Next lowest priority 10 Next highest priority 11 Highest priority
24–30	—	Reserved
31	SWR	Software reset. Writing 1 to this bit causes a global software reset. Upon completion of the reset, this bit is automatically cleared. 0 Don't reset 1 Global reset
32–39	CHN3_EU_PR_CNT	Channel 3 EU priority counter. This counter is used by the controller to determine when Channel 3 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. <b>Note:</b> If set to zero, the CHN4_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHN4_EU_PR_CTR must also be set to a different, non-zero value.
40–47	CHN4_EU_PR_CNT	Channel 4 EU priority counter. This counter is used by the controller to determine when Channel 4 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. <b>Note:</b> If set to zero, the CHN3_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHN3_EU_PR_CTR must also be set to a different, non-zero value.

Table 12-62. Master Control Register (MCR) Field Descriptions (continued)

Bits	Name	Description
48–55	CHN3_BUS_PR_CNT	Channel 3 bus priority counter. This counter is used by the controller to determine when Channel 3 has been denied access to the bus long enough to warrant immediate elevation to top priority. <b>Note:</b> If set to zero, the CHN4_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN4_BUS_PR_CTR must also be set to a different, non-zero value.
56–63	CHN4_BUS_PR_CNT	Channel 4 bus priority counter. This counter is used by the controller to determine when Channel 4 has been denied access to a needed on-chip resource long enough to warrant immediate elevation to top priority. <b>Note:</b> If set to zero, the CHN3_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN3_BUS_PR_CTR must also be set to a different, non-zero value.

## 12.7 Power-Saving Mode

The SEC may be disabled by setting DEVDISR[SEC]. (See [Section 19.4.1.12](#), “Device Disable Register (DEVDISR),” for more information on this register.). The clocks to the SEC are active by default. The SEC should not be enabled/disabled during normal operation.



# Chapter 13

## DUART

This chapter describes the dual universal asynchronous receiver/transmitters (DUART). It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

### 13.1 Overview

The DUART consists of two universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the core complex bus (CCB) clock. The DUART programming model is compatible with the PC16552D.

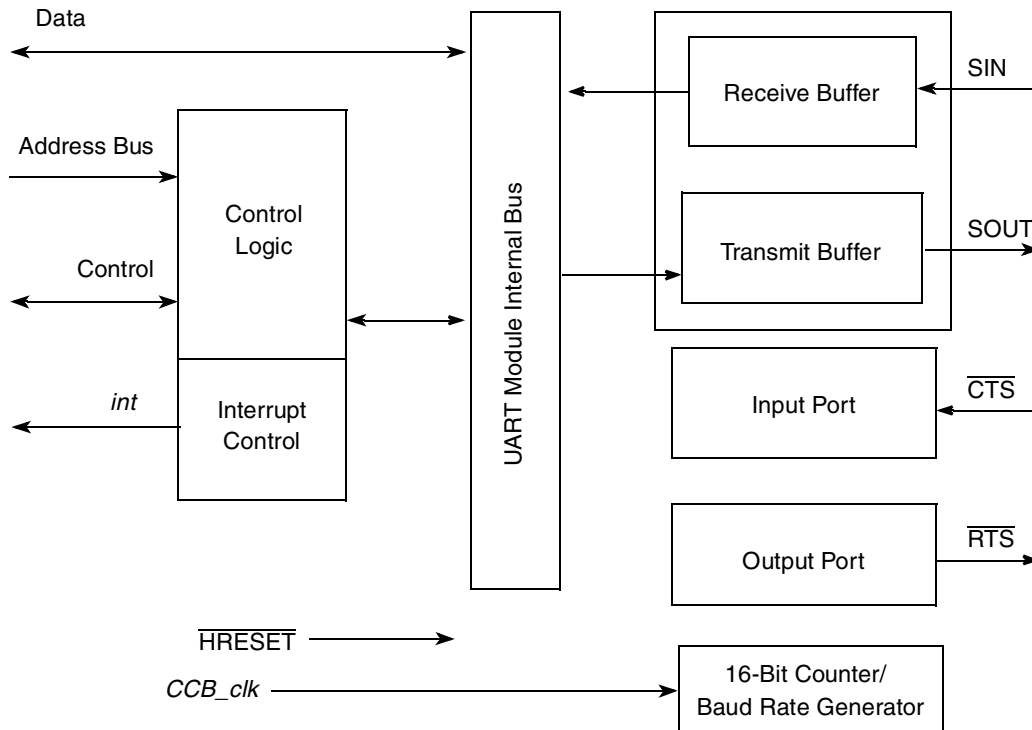
The UART interface is point to point, meaning that only two UART devices are attached to the connecting signals. As shown in [Figure 13-1](#), each UART module consists of the following:

- Receive and transmit buffers
- Clear to send ( $\overline{\text{CTS}}$ ) input port and request to send ( $\overline{\text{RTS}}$ ) output port for data flow control
- 16-bit counter for baud rate generation
- Interrupt control logic

#### 13.1.1 Features

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the CCB clock by 1 to  $(2^{16} - 1)$  and generate a 16x clock for the transmitter and receiver engines



**Figure 13-1. UART Block Diagram**

- Clear to send ( $\overline{\text{CTS}}$ ) and ready to send ( $\overline{\text{RTS}}$ ) modem control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

### 13.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a start bit, parity (if any), stop bits, and transfers the assembled character (with start, stop, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.



## 13.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

### 13.2.1 Signal Overview

Table 13-1 summarizes the DUART signals. Note that although the actual device signal names are prepended with the UART\_ prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

**Table 13-1. DUART Signal Overview**

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[0:1]	I	2	1	Serial in data UART0 and UART1
UART_SOUT[0:1]	O	2	1	Serial out data UART0 and UART1
$\overline{\text{UART\_CTS}}[0:1]$	I	2	1	Clear to send UART0 and UART1
$\overline{\text{UART\_RTS}}[0:1]$	O	2	1	Request to send UART0 and UART1

### 13.2.2 Detailed Signal Descriptions

The DUART signals are described in detail in Table 13-2.

**Table 13-2. DUART Signals—Detailed Signal Descriptions**

Signal	I/O	Description
UART_SIN[0:1]	I	Serial data in. Data is received on the receivers of UART0 and UART1 through its respective serial data input signal, with the least-significant bit received first.
		<b>State Meaning</b> Asserted/Negated—Represents the data being received on the UART interface.
		<b>Timing</b> Assertion/Negation—An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to sample the data on SIN.
UART_SOUT[0:1]	O	Serial data out. The serial data output signals for the UART0 and UART1 are set ('mark' condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on these signals, with the least significant bit transmitted first.
		<b>State Meaning</b> Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		<b>Timing</b> Assertion/Negation— An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to update and drive the data on SOUT.
$\overline{\text{UART\_CTS}}[0:1]$	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective $\overline{\text{RTS}}$ outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal.
		<b>State Meaning</b> Asserted/Negated—Represent the clear to send condition for their respective UART.
		<b>Timing</b> Assertion/Negation—Sampled at the rising edge of every CCB clock.

Table 13-2. DUART Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
UART_RTS[0:1]	O	Request to send. UART_RTSx are active-low output signals that can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send (CTS) input of a transmitter, this signal can be used to control serial data flow.
		<b>State Meaning</b> Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		<b>Timing</b> Assertion/Negation—Updated and driven at the rising edge of every CCB clock.

### 13.3 Memory Map/Register Definition

Table 13-3 lists the DUART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 13-3.

There are two complete sets of DUART registers (one for UART0 and one for UART1). The two UARTs on the device are identical, except that the registers for UART0 are located at offset 0x4500 (local), and the registers for UART1 are located at offset 0x4600 (local). Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART0 or UART1.

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to Section 13.3.1.7, “Line Control Registers (ULCR0, ULCR1),” for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Reads and writes to these registers must be byte-wide operations. Table 13-3 provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 13-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
<b>Block Base Address: 0x0_4000</b>				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	<a href="#">13.3.1.1/13-6</a>
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	<a href="#">13.3.1.2/13-6</a>
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	<a href="#">13.3.1.4/13-9</a>
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	<a href="#">13.3.1.5/13-9</a>
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	<a href="#">13.3.1.6/13-11</a>
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	<a href="#">13.3.1.12/13-17</a>
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	<a href="#">13.3.1.7/13-12</a>
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	<a href="#">13.3.1.8/13-14</a>
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	<a href="#">13.3.1.9/13-15</a>
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	<a href="#">13.3.1.10/13-16</a>
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	<a href="#">13.3.1.11/13-17</a>
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	<a href="#">13.3.1.13/13-18</a>
0x600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	<a href="#">13.3.1.1/13-6</a>
0x600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	<a href="#">13.3.1.2/13-6</a>
0x600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	<a href="#">13.3.1.4/13-9</a>
0x601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	<a href="#">13.3.1.5/13-9</a>
0x602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	<a href="#">13.3.1.6/13-11</a>
0x602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	<a href="#">13.3.1.12/13-17</a>
0x603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	<a href="#">13.3.1.7/13-12</a>
0x604	UMCR—ULCR[DLAB] = x UART1 modem control register	R/W	0x00	<a href="#">13.3.1.8/13-14</a>
0x605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	<a href="#">13.3.1.9/13-15</a>
0x606	UMSR—ULCR[DLAB] = x UART1 modem status register	R	0x00	<a href="#">13.3.1.10/13-16</a>
0x607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	<a href="#">13.3.1.11/13-17</a>
0x610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	<a href="#">13.3.1.13/13-18</a>

## 13.3.1 Register Descriptions

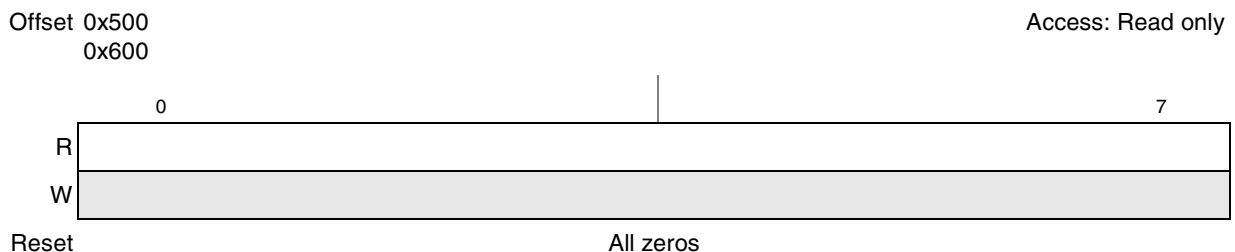
The following sections describe the UART0 and UART1 registers.

### 13.3.1.1 Receiver Buffer Registers (URBR0, URBR1) (ULCR[DLAB] = 0)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, [Section 13.3.1.9, “Line Status Registers \(ULSR0, ULSR1\).”](#) [Figure 13-3](#) shows the receiver buffer registers. Note that these registers have same offset as the UTHR.

[Figure 13-2](#) shows the bits in the URBRs.



**Figure 13-2. Receiver Buffer Registers (URBR0, URBR1)**

[Table 13-4](#) describes the fields of URBR.

**Table 13-4. URBR Field Descriptions**

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus (read only)

### 13.3.1.2 Transmitter Holding Registers (UTHR0, UTHR1) (ULCR[DLAB] = 0)

A write to these 8-bit registers causes the UART devices to transfer 5–8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR will be the first byte onto the bus. UDSR[ $\overline{\text{TXRDY}}$ ] indicates when the FIFO is full. Refer to the [Table 13-21](#) and the [Table 13-22](#) for more details.

Figure 13-3 shows the bits in the UTHR.

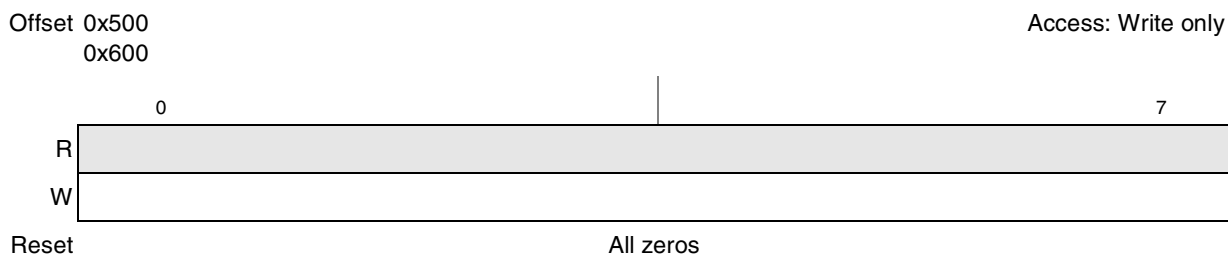


Figure 13-3. Transmitter Holding Registers (UTHR0, UTHR1)

Table 13-5 describes the fields of UTHR.

Table 13-5. UTHR Field Descriptions

Bits	Name	Description
0–7	DATA	Data that is written to UTHR (write only)

### 13.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (ULCR[DLAB] = 1)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency / (16 × [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency / desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 13-8.

Figure 13-4 shows the bits in the UDMBs.

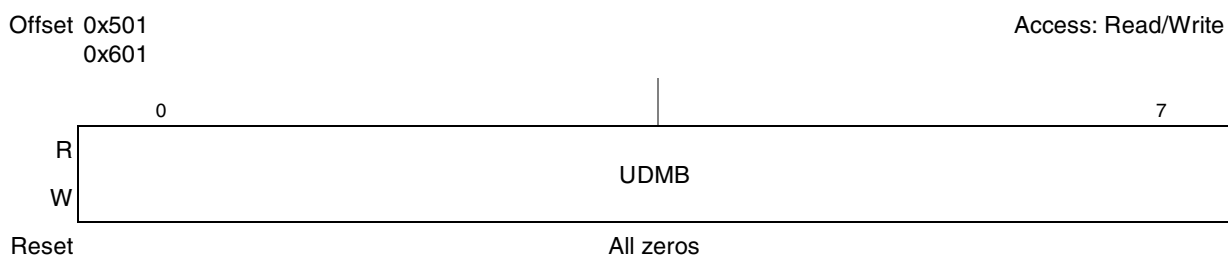


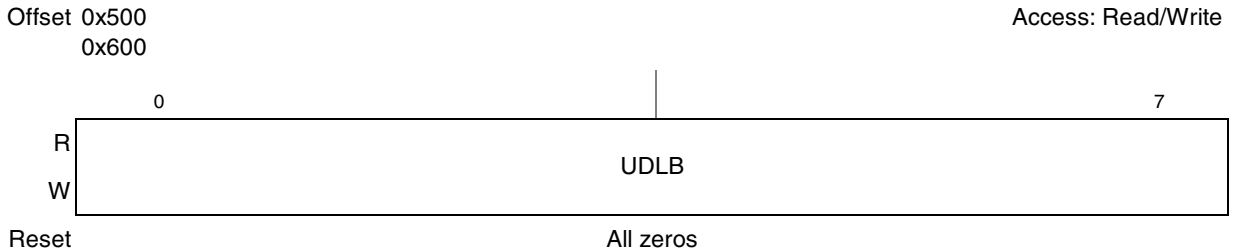
Figure 13-4. Divisor Most Significant Byte Registers (UDMB0, UDMB1)

Table 13-6 describes the fields of UDMB registers.

Table 13-6. UDMB Field Descriptions

Bits	Name	Description
0–7	UDMB	Divisor most-significant byte

Figure 13-5 shows the bits in the UDLBs.



**Figure 13-5. Divisor Least Significant Byte Registers (UDLB0, UDLB1)**

Table 13-7 describes the fields of UDLB registers.

**Table 13-7. UDLB Field Descriptions**

Bits	Name	Description
0–7	UDLB	Divisor least-significant byte. This is concatenated with UDMB.

Table 13-8 shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible. Note that because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

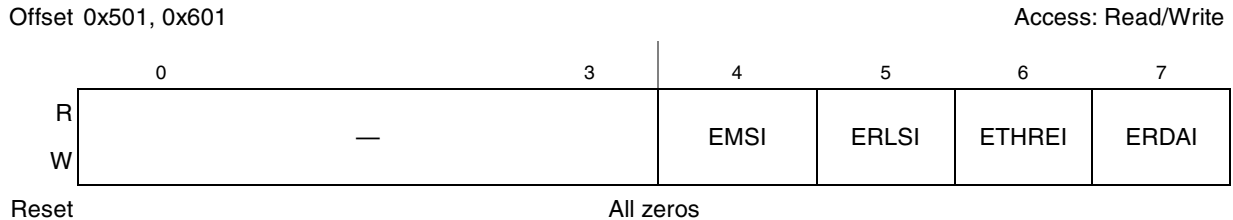
**Table 13-8. Baud Rate Examples**

Target Baud Rate (Decimal)	Divisor		Input Clock (CCB) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1736	6C8	266	9600.61444	0.0064
19,200	868	364	266	19,201.22888	0.0064
38,400	434	1B2	266	38,402.45776	0.0064
57,600	289	121	266	57,670.12673	0.1217
115,200	145	91	266	114,942.52844	0.2235
230,400	72	48	266	231,481.48090	0.4694
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
57,600	362	16A	333	57,550.64457	0.0857
115,200	181	B5	333	115,101.28913	0.0857
230,400	90	5A	333	231,481.48148	0.4694

### 13.3.1.4 Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Figure 13-6 shows the bits in the UIER.



**Figure 13-6. Interrupt Enable Register (UIER)**

Table 13-9 describes the fields of UIER.

**Table 13-9. UIER Field Descriptions**

Bits	Name	Description
0–3	—	Reserved.
4	EMSI	Enable modem status interrupt. 0 Mask interrupts caused by UMSR[DCTS] being set 1 Enable and assert interrupts when the clear-to-send bit in the UART modem status register (UMSR) changes state
5	ERLSI	Enable receiver line status interrupt. 0 Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1 Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set
6	ETHREI	Enable transmitter holding register empty interrupt. 0 Mask interrupt when ULSR[THRE] is set 1 Enable and assert interrupts when ULSR[THRE] is set
7	ERDAI	Enable received data available interrupt. 0 Mask interrupt when new receive data is available or receive data time out has occurred 1 Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in the FIFO mode

### 13.3.1.5 Interrupt ID Registers (UIR0, UIR1) (ULCR[DLAB] = 0)

The UIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIR. The four levels of interrupt conditions in order of priority are:

1. Receiver line status
2. Received data ready/character time-out

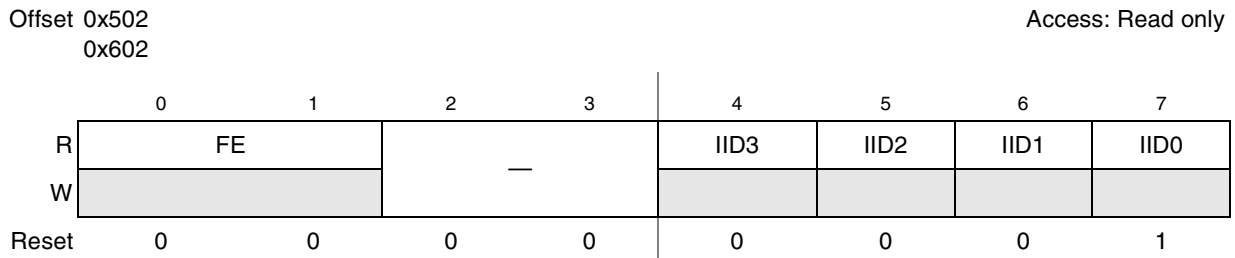
## DUART

3. Transmitter holding register empty
4. Modem status

See [Table 13-11](#) for more details.

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

[Figure 13-7](#) shows the bits in the UIIR.



**Figure 13-7. Interrupt ID Registers (UIIR)**

[Table 13-10](#) describes the fields of the UIIR.

**Table 13-10. UIIR Field Descriptions**

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in <a href="#">Table 13-11</a> . IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5–6	IID2–1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in <a href="#">Table 13-11</a> .
7	IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

The bits contained in the UIIR registers are described in [Table 13-11](#).

**Table 13-11. UIIR IID Bits Summary**

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	—	—	—	—
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.



Table 13-11. UIIR IID Bits Summary (continued)

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Read the UMSR.

### 13.3.1.6 FIFO Control Registers (UFCR0, UFCR1) (ULCR[DLAB] = 0)

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

Figure 13-8 shows the bits in the UFCRs.

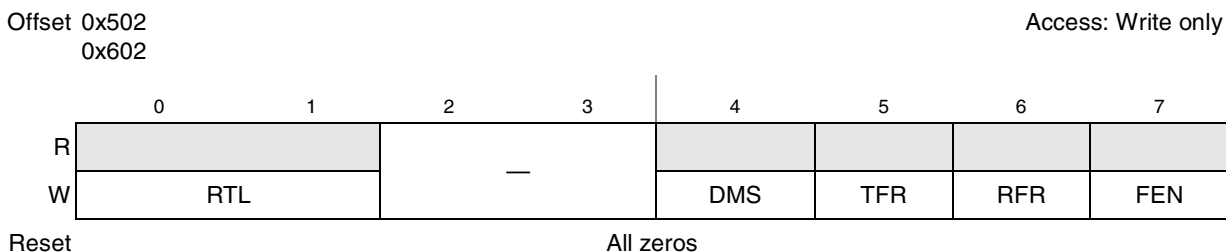


Figure 13-8. FIFO Control Registers (UFCR0, UFCR1)

Table 13-12 describes the fields of the UFCRs.

**Table 13-12. UFCR Field Descriptions**

Bits	Name	Description
0–1	RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved
4	DMS	DMA mode select. See <a href="#">Section 13.4.5.2, “DMA Mode Select,”</a> for more information. 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5	TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

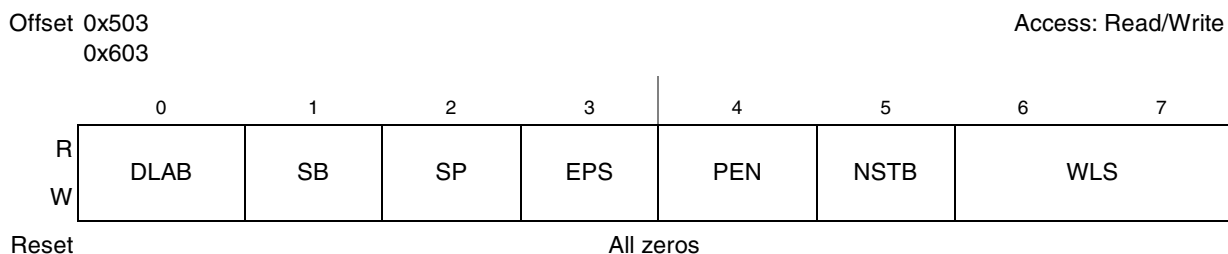
### 13.3.1.7 Line Control Registers (ULCR0, ULCR1)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See [Table 13-14](#) for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 13-9 shows the bits in the ULCRs.



**Figure 13-9. Line Control Register (ULCR)**

Table 13-13 describes the fields of the ULCRs.

**Table 13-13. ULCR Field Descriptions**

Bits	Name	Description
0	DLAB	Divisor latch access bit. 0 Access to all registers except UDLB, UAFR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
1	SB	Set break. 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2	SP	Stick parity. 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 13-14 for more information. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4	PEN	Parity enable. 0 No parity generation and checking 1 Generate parity bit as a transmitter, and check parity as a receiver
5	NSTB	Number of STOP bits. 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1½ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6-7	WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

**Table 13-14. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]**

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

### 13.3.1.8 Modem Control Registers (UMCR0, UMCR1)

The UMCRs control the interface with the external peripheral device on the UART bus.

Figure 13-10 shows the bits in the UMCRs

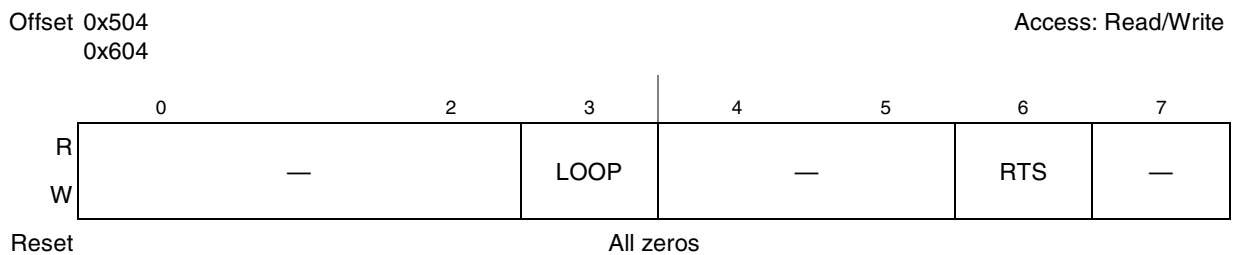
**Figure 13-10. Modem Control Register (UMCR)**

Table 13-15 describes the fields of UMCRs.

**Table 13-15. UMCR Field Descriptions**

Bits	Name	Description
0–2	—	Reserved.
3	LOOP	Local loopback mode. 0 Normal operation 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5	—	Reserved.
6	RTS	Ready to send. 0 Negates corresponding $\overline{\text{UART\_RTS}}$ output 1 Assert corresponding $\overline{\text{UART\_RTS}}$ output. Informs external modem or peripheral that the UART is ready for sending/receiving data
7	—	Reserved.

### 13.3.1.9 Line Status Registers (ULSR0, ULSR1)

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

Figure 13-11 shows the bits in the ULSRs.

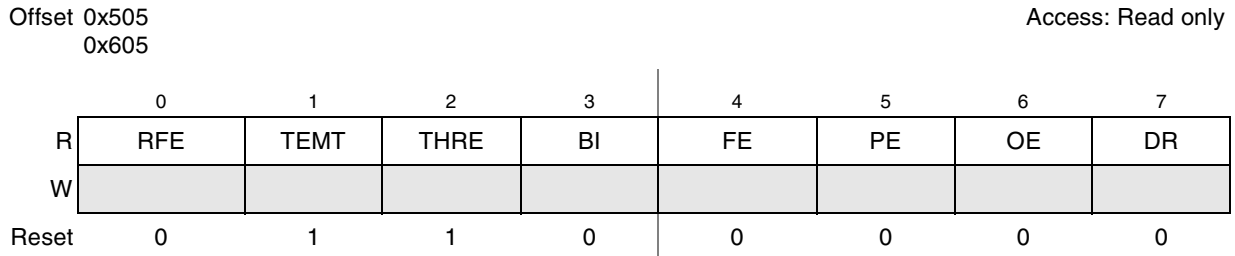


Figure 13-11. Line Status Register (ULSR)

Table 13-16 describes the fields of the ULSRs.

Table 13-16. ULSR Field Descriptions

Bits	Name	Description
0	RFE	Receiver FIFO error. 0 This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1	TEMT	Transmitter empty. 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty. 0 The UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt. 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4	FE	Framing error. 0 This bit is cleared when ULSR is read or when a new bit character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then will receive the following new data.

Table 13-16. ULSR Field Descriptions (continued)

Bits	Name	Description
5	PE	Parity error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.
6	OE	Overrun error. 0 This bit is cleared when ULSR is read. 1 Before the URBR is read, the URBR was overwritten with a new character. The old character is lost. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready. 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

### 13.3.1.10 Modem Status Registers (UMSR0, UMSR1)

The UMSRs track the status of the modem (or external peripheral device) clear to send ( $\overline{CTS}$ ) signal for the corresponding UART.

Figure 13-12 shows the bits in the UMSRs.

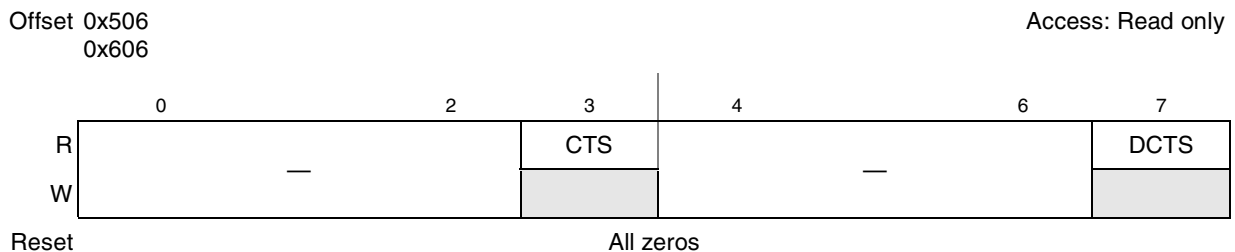


Figure 13-12. Modem Status Register (UMSR)

Table 13-17 describes the fields of the UMSRs.

Table 13-17. UMSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved.
3	CTS	Clear to send. Represents the inverted value of the $\overline{CTS}$ input pin from the external peripheral device 0 Corresponding $\overline{CTS}_n$ is negated 1 Corresponding $\overline{CTS}_n$ is asserted. The modem or peripheral device is ready for data transfers.
4–6	—	Reserved.
7	DCTS	Clear to send. 0 No change on the corresponding $\overline{CTS}_n$ signal since the last read of UMSR[CTS] 1 The $\overline{CTS}_n$ value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMS] is set to detect this condition

### 13.3.1.11 Scratch Registers (USCR0, USCR1)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Figure 13-13 shows the bits in USCRs.



Figure 13-13. Scratch Register (USCR)

Table 13-18 describes the fields of the USCRs.

Table 13-18. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

### 13.3.1.12 Alternate Function Registers (UAFR0, UAFR1) (ULCR[DLAB] = 1)

The UAFRs give software the ability to write to both UART0 and UART1 registers simultaneously with the same write operation. The UAFRs also provide a means for the device's performance monitor to track the baud clock.

Figure 13-14 shows the bits in the UAFRs.

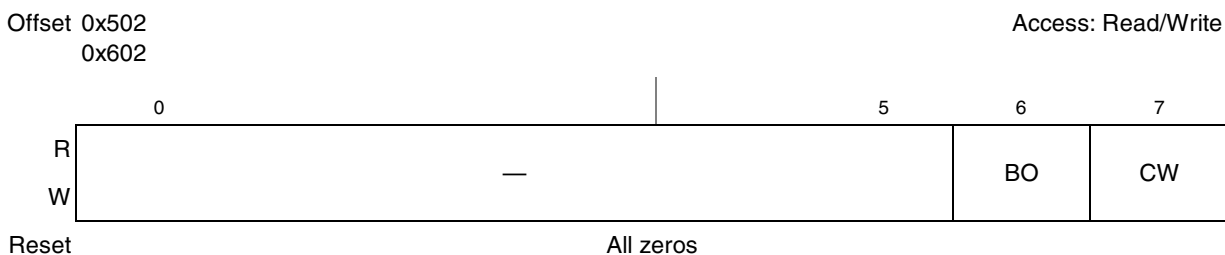


Figure 13-14. Alternate Function Register (UAFR)

Table 13-19 describes the fields of the UAFRs.

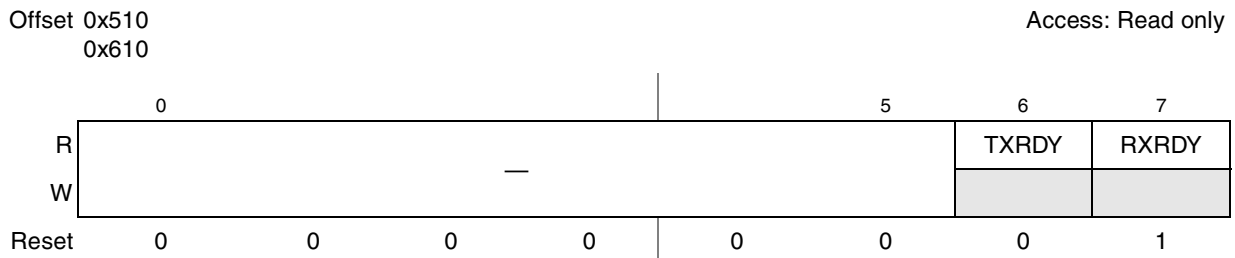
**Table 13-19. UAFR Field Descriptions**

Bits	Name	Description
0–5	—	Reserved.
6	BO	Baud clock select. 0 The baud clock is not gated off. 1 The baud clock is gated off.
7	CW	Concurrent write enable. 0 Disables writing to both UART0 and UART1 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART0 is also a write to the corresponding register in UART1 and vice versa. The user needs to ensure that the LCR[DLAB] of both UARTs are in the same state before executing a concurrent write to register addresses 0x500, 0x501 and 0x502.

### 13.3.1.13 DMA Status Registers (UDSR0, UDSR1)

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Figure 13-15 shows the bits in UDSRs.



**Figure 13-15. DMA Status Register (UDSR)**

Table 13-20 describes the fields of the UDSRs.

**Table 13-20. UDSR Field Descriptions**

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 13-22. 1 This bit is set, as shown in Table 13-21.
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 13-24. 1 This bit is set, as shown in Table 13-23.



Table 13-21. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 13-22. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear when the transmitter FIFO is not yet full.

Table 13-23. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 13-24. UDSR[RXRDY] Cleared

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

## 13.4 Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock signal.

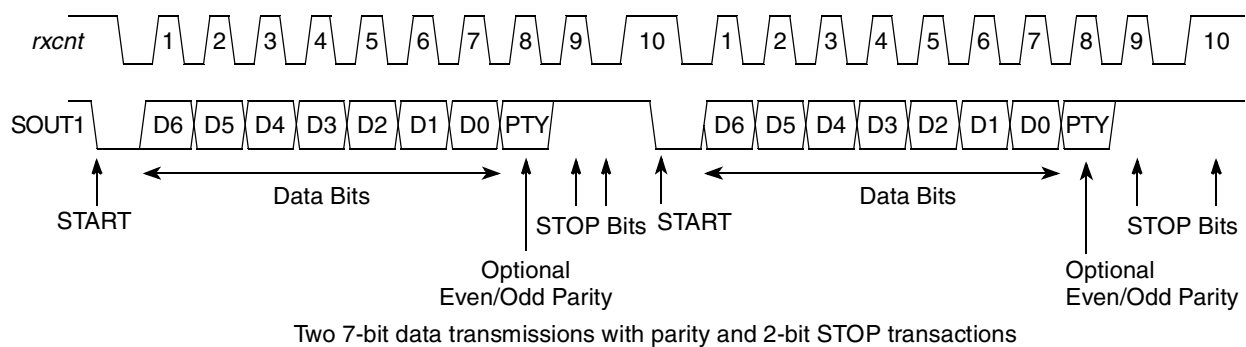
The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see [Section 13.4.5, “FIFO Mode.”](#) The transmitting registers convert the data to a serial bit stream, by inserting the appropriate START, STOP, and optional parity bits. Finally, the registers output a

composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt-driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

### 13.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in [Figure 13-16](#). Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



**Figure 13-16. UART Bus Interface Transaction Protocol Example**

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

#### 13.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal.

[Figure 13-16](#) shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in the UART line control register (ULCR). When the bus is idle, SOUT is high.

### 13.4.1.2 Data Transfer

Each data transfer contains 5–8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5–8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

### 13.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see [Section 13.3.1.7, “Line Control Registers \(ULCR0, ULCR1\).”](#) Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data a parity error can occur if an unexpected parity value is detected. (See [Section 13.3.1.9, “Line Status Registers \(ULSR0, ULSR1\).”](#))

### 13.4.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

## 13.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the CCB clock input and dividing the input by any divisor from 1 to  $2^{16} - 1$ .

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

$$\text{Baud rate} = (1/16) \times (\text{CCB clock frequency}/\text{divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

### 13.4.3 Local Loopback Mode

Local loopback mode is provided for diagnostic testing. The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the modem control register UMCR[RTS] is internally tied to the modem status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The  $\overline{\text{CTS}}$  (input signal) is disconnected,  $\overline{\text{RTS}}$  is internally connected to  $\overline{\text{CTS}}$ , and the  $\overline{\text{RTS}}$  (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loopback mode the transmit and receive data paths of the DUART can be verified. Note that in local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

### 13.4.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

#### 13.4.4.1 Framing Error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

#### 13.4.4.2 Parity Error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

#### 13.4.4.3 Overrun Error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

### 13.4.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs

and set the FIFO receiver trigger level `UFCR[RTL]` to control the received data available interrupt `UIER[ERDAI]`.

The `UFCR` also selects the type of DMA signaling. The `UDSR[RXRDY]` indicates the status of the receiver FIFO. The DMA status registers (`UDSR[TXRDY]`) indicate when the transmitter FIFO is full. When in FIFO mode, data written to `UTHR` is placed into the transmitter FIFO. The first byte written to `UTHR` is the first byte onto the UART bus.

### 13.4.5.1 FIFO Interrupts

In FIFO mode, the `UIER[ERDAI]` is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through `UIER[ERDAI]`). See [Section 13.3.1.4, “Interrupt Enable Register \(UIER\) \(ULCR\[DLAB\] = 0\),”](#) for more details on interrupt enables.

The interrupt ID register (`UIIR`) indicates if the FIFOs are enabled. Interrupt ID3 `UIIR[IID3]` bit is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by `UIIR[IID]`) is cleared when the `URBR` is read. See [Section 13.3.1.5, “Interrupt ID Registers \(UIR0, UIR1\) \(ULCR\[DLAB\] = 0\),”](#) for more information.

The `UIIR[FE]` bits indicate if FIFO mode is enabled.

### 13.4.5.2 DMA Mode Select

The `UDSR[RXRDY]` bit reflects the status of the receiver FIFO or `URBR`. In mode 0 (`UFCR[DMS]` is cleared), `UDSR[RXRDY]` is cleared when there is at least one character in the receiver FIFO or `URBR` and it is set when there are no more characters in the receiver FIFO or `URBR`. This occurs regardless of the setting of the `UFCR[FEN]` bit. In mode 1 (`UFCR[DMS]` and `UFCR[FEN]` are set), `UDSR[RXRDY]` is cleared when the trigger level or a time-out has been reached and it is set when there are no more characters in the receiver FIFO.

The `UDSR[TXRDY]` bit reflects the status of the transmitter FIFO or `UTHR`. In mode 0 (`UFCR[DMS]` is cleared), `UDSR[TXRDY]` is cleared when there are no characters in the transmitter FIFO or `UTHR` and it is set after the first character is loaded into the transmitter FIFO or `UTHR`. This occurs regardless of the setting of the `UFCR[FEN]` bit. In mode 1 (`UFCR[DMS]` and `UFCR[FEN]` are set), `UDSR[TXRDY]` is cleared when there are no characters in the transmitter FIFO or `UTHR` and it is set when the transmitter FIFO is full.

See [Section 13.3.1.13, “DMA Status Registers \(UDSR0, UDSR1\),”](#) for a complete description of the `UDSR[RXRDY]` and `UDSR[TXRDY]` bits.

### 13.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 0 (`UIIR[0]`), is cleared. The interrupt enable register (`UIER`) is used to mask specific interrupt types. For more details refer to the description of `UIER` in [Section 13.3.1.4, “Interrupt Enable Register \(UIER\) \(ULCR\[DLAB\] = 0\).”](#)

When the interrupts are disabled in UIER, polling software can not use UIIR[0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the modem status (UMSR) registers. UIIR[0] can be used for polling if the interrupts are enabled in UIER.

## **13.5 DUART Initialization/Application Information**

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01X1.)
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external modem or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR if the interrupts generated by the DUART are masked.

# Chapter 14

## Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), SDRAM machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes an initialization and applications information section with many specific examples of its use.

### 14.1 Introduction

Figure 14-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and SDRAM controller.

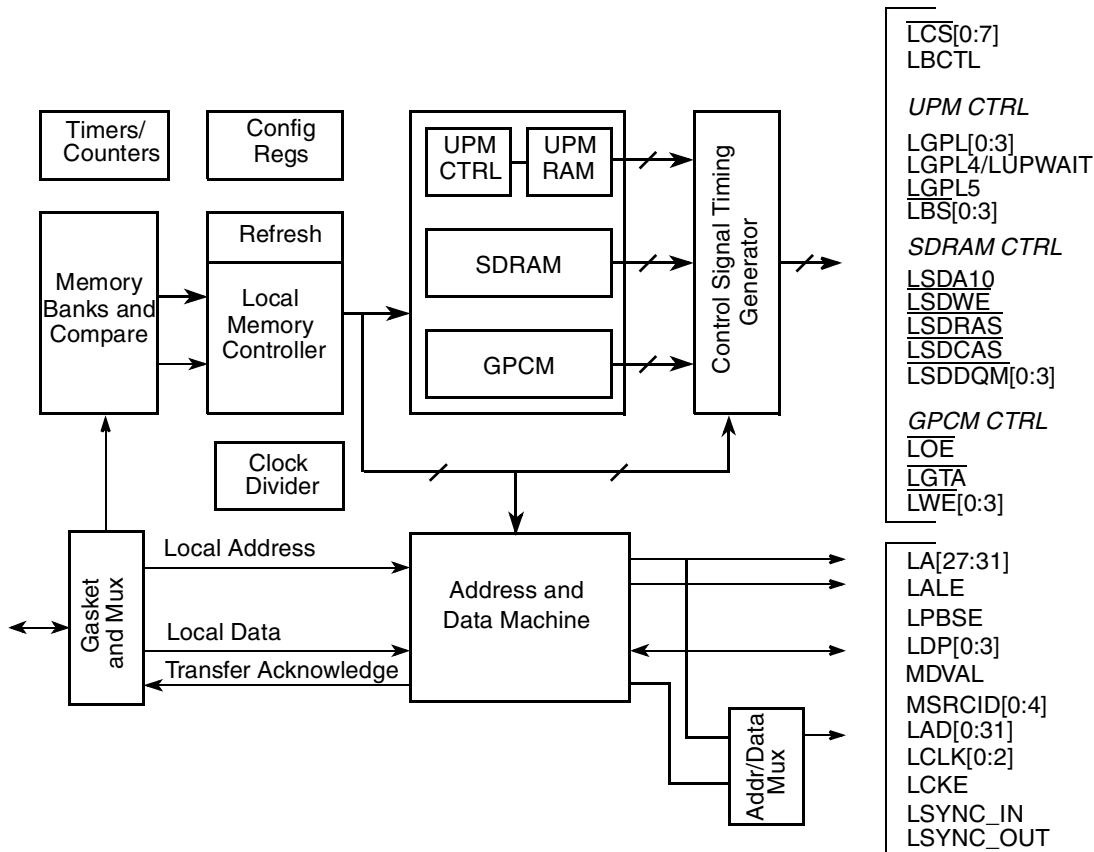


Figure 14-1. Local Bus Controller Block Diagram

## 14.1.1 Overview

The main component of the LBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a GPCM, and up to three UPMs. As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LAL) allows multiplexing of addresses with data signals to reduce the device signal count.

The LBC also includes a number of data checking and protection features such as data parity generation and checking, write protection and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

## 14.1.2 Features

The LBC main features are as follows:

- Memory controller with eight memory banks
  - 34-bit<sup>1</sup> address decoding with mask
  - Variable memory block sizes (32 Kbytes to 4 Gbytes)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Up to 256-byte bursts, arbitrarily aligned
  - Automatic segmentation of large transactions
  - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
  - Write-protection capability
  - Atomic operation
  - Parity byte-select
- SDRAM machine
  - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
  - Supports up to four concurrent open pages per device
  - Supports SDRAM port size of 32, 16, and 8 bits
  - Supports external address and/or command lines buffering
- General-purpose chip-select machine (GPCM)
  - Compatible with SRAM, EPROM, FEPRM, and peripherals
  - Global (boot) chip-select available at system reset
  - Boot chip-select support for 8-, 16-, 32-bit devices
  - Minimum 3-clock access to external devices

1. Refers to the logical address space of the LBC. Once the address is decoded by the LBC, the 34-bit address becomes the right-most 34 bits of the 36-bit physical address space.



- Four byte-write-enable signals ( $\overline{\text{LWE}}[0:3]$ )
- Output enable signal ( $\overline{\text{LOE}}$ )
- External access termination signal ( $\overline{\text{LGTA}}$ )
- Three user-programmable machines (UPMs)
  - Programmable-array-based machine controls external signal timing with a granularity of up to one-quarter of an external bus clock period
  - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access.
  - UPM refresh timer runs a user-specified control signal pattern to support refresh
  - User-specified control-signal patterns can be initiated by software
  - Each UPM can be defined to support DRAM devices with depths of 64, 128, 256, and 512 Kbytes, and 1, 2, 4, 8, 16, 32, 64, 128, and 256 Mbytes
  - Support for 8-, 16-, 32-bit devices
  - Page mode support for successive transfers within a burst
  - Internal address multiplexing supporting 64-, 128-, 256-, and 512-Kbyte, and 1-, 2-, 4-, 8-, 16-, 32-, 64-, 128-, and 256-Mbyte page banks
- Optional monitoring of transfers between local bus internal masters and local bus slaves (local bus error reporting)
- Support for phase-locked loop (PLL) with software-configurable bypass for low frequency bus clocks

### 14.1.3 Modes of Operation

The LBC provides one GPCM, one SDRAM machine, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the LBC, the memory address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, SDRAM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus, with the LBC in GPCM, SDRAM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction.

#### 14.1.3.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 4, 8, and 16 between the faster system (CCB) clock and the slower external bus clock ( $\text{LCLK}[0:2]$ ). This ratio is software programmable through the clock ratio register ( $\text{LCRR}[\text{CLKDIV}]$ ). In addition to establishing the frequency of the external local bus clock,  $\text{CLKDIV}$  also affects the resolution of signal timing shifts in GPCM mode and the interpretation of UPM array words in UPM mode. The bus clock is driven identically onto signals  $\text{LCLK}[0:2]$  to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

### 14.1.3.2 Source ID Debug Mode

In debug mode, the LBC provides the ID of a transaction source on external device signals. This mode is enabled on power-on reset, as described in [Section 21.4.3, “Local Bus Interface Debug.”](#) When placed in this mode, the 5-bit internal ID of the current transaction source appears on MSRCID[0:4] whenever valid address or data is available on the LBC external signals. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID signals at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (MDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on MSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:31]. Note that in SDRAM mode the address vector contains the full address as {row, bank, column, lsbs} where row corresponds to the same row address for the given column address and lsbs are the unconnected lsbs of the address for a given port size.
- If a valid source ID is detected on MSRCID[0:4] and MDVAL is asserted, valid data may be latched from LAD[0:31].

### 14.1.4 Power-Down Mode

The LBC can enter a power-down mode when the system stops the internal (system) clock to the block by using a handshake protocol initiated by the DEVDISR[LBC] setting in the global utilities block. On entering power-down mode, the LBC places any SDRAM devices, if used, in self-refresh mode before the bus clock is stopped. The LBC also allows the PLL sufficient time to recover following the reapplication of the system clock.

Once the LBC has been put into power-down mode, the only way to exit from this mode is through HRESET.

## 14.2 External Signal Descriptions

[Table 14-1](#) contains a list of external signals related to the LBC and summarizes their function. I/O impedance of designated local bus signals is determined by PORIMPSCR, as described in [Section 19.4.1.3, “POR I/O Impedance Status and Control Register \(PORIMPSCR\).”](#)

**Table 14-1. Signal Properties—Summary**

Name	Number of Signals	Direction	Function
LALE	1	Output	External address latch enable
$\overline{\text{LCS}}$	8	Output	Chip selects
$\overline{\text{LWE}}_n$ / $\overline{\text{LSDDQM}}_n$ / $\overline{\text{LBS}}_n$	4	Output Output Output	GPCM mode: write enable SDRAM mode: byte lane data mask UPM mode: byte (lane) select
LSDA10/ LGPL0	1	Output Output	SDRAM mode: row address bit/command bit UPM mode: general-purpose line 0
$\overline{\text{LSDWE}}$ / LGPL1	1	Output Output	SDRAM mode: write enable UPM mode: general-purpose line 1

Table 14-1. Signal Properties—Summary (continued)

Name	Number of Signals	Direction	Function
$\overline{\text{LOE}}$ / $\overline{\text{LSDRAS}}$ / $\overline{\text{LGPL2}}$	1	Output Output Output	GPCM mode: output enable SDRAM mode: row address strobe UPM mode: general-purpose line 2
$\overline{\text{LSDCAS}}$ / $\overline{\text{LGPL3}}$	1	Output Output	SDRAM mode: column address strobe UPM mode: general-purpose line 3
$\overline{\text{LGTA}}$ / $\overline{\text{LGPL4}}$ / $\overline{\text{LUPWAIT}}$ / $\overline{\text{LPBSE}}$	1	Input Output Input Output	GPCM mode: transaction termination UPM mode: general-purpose line 4 UPM mode: external device wait Local bus parity byte select
LGPL5	1	Output	UPM mode: general-purpose line 5
LBCTL	1	Output	Data buffer control
LA[27:31]	5	Output	Local bus non-multiplexed address lsbs
LAD[0:31]	32	Input/Output	Multiplexed address/data bus
LDP	4	Input/Output	Local bus data parity
LCKE	1	Output	Local bus clock enable
LCLK[0:2]	3	Output	Local bus clocks
LSYNC_IN	1	Input	PLL synchronize input
LSYNC_OUT	1	Output	PLL synchronize output
MDVAL	1	Output	In LBC debug mode: local bus data valid
MSRCID	5	Output	In LBC debug mode: local bus source ID

Table 14-2 contains detailed external signal descriptions for the LBC.

Table 14-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device signals.
		<b>State Meaning</b> Asserted/Negated—LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. The exact timing of the negation of LALE is controlled by the LBCR[AHD] field. Note that no other control signals are asserted during the assertion of LALE.
$\overline{\text{LCS}}[0:7]$	O	Chip selects. Eight chip selects are provided which are mutually exclusive.
		<b>State Meaning</b> Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. $\overline{\text{LCS}}[0:7]$ are provided on a per-bank basis with $\overline{\text{LCS}}_0$ corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.

Table 14-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description				
$\overline{\text{LWE}}[0:3]/$ $\overline{\text{LSDDQM}}[0:3]/$ $\overline{\text{LBS}}[0:3]$	O	<p>GPCM write enable/SDRAM data mask/UPM byte select. These signals select or validate each byte lane of the data bus. For banks with port sizes of 32 bits (as set by <math>\text{BR}_n[\text{PS}]</math>), all four signals are defined. For a 16-bit port size, only bits 0:1 are defined; and for an 8-bit port size, bit 0 is the only defined signal. The least-significant address bits of each access also determine which byte lanes are considered valid for a given data transfer.</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—For GPCM operation, <math>\overline{\text{LWE}}[0:3]</math> assert for each byte lane enabled for writing. For SDRAM operation, <math>\overline{\text{LSDDQM}}[0:3]</math> function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. <math>\overline{\text{LSDDQM}}[0:3]</math> are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. <math>\overline{\text{LBS}}[0:3]</math> are programmable byte-select signals in UPM mode. See Section 14.4.4.4, “RAM Array,” for programming details about <math>\overline{\text{LBS}}[0:3]</math>.</td> </tr> <tr> <td><b>Timing</b></td> <td>Assertion/Negation—See Section 14.4.2, “General-Purpose Chip-Select Machine (GPCM),” for details regarding the timing of <math>\overline{\text{LWE}}[0:3]</math>.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—For GPCM operation, $\overline{\text{LWE}}[0:3]$ assert for each byte lane enabled for writing. For SDRAM operation, $\overline{\text{LSDDQM}}[0:3]$ function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. $\overline{\text{LSDDQM}}[0:3]$ are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. $\overline{\text{LBS}}[0:3]$ are programmable byte-select signals in UPM mode. See Section 14.4.4.4, “RAM Array,” for programming details about $\overline{\text{LBS}}[0:3]$ .	<b>Timing</b>	Assertion/Negation—See Section 14.4.2, “General-Purpose Chip-Select Machine (GPCM),” for details regarding the timing of $\overline{\text{LWE}}[0:3]$ .
<b>State Meaning</b>	Asserted/Negated—For GPCM operation, $\overline{\text{LWE}}[0:3]$ assert for each byte lane enabled for writing. For SDRAM operation, $\overline{\text{LSDDQM}}[0:3]$ function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. $\overline{\text{LSDDQM}}[0:3]$ are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. $\overline{\text{LBS}}[0:3]$ are programmable byte-select signals in UPM mode. See Section 14.4.4.4, “RAM Array,” for programming details about $\overline{\text{LBS}}[0:3]$ .					
<b>Timing</b>	Assertion/Negation—See Section 14.4.2, “General-Purpose Chip-Select Machine (GPCM),” for details regarding the timing of $\overline{\text{LWE}}[0:3]$ .					
$\text{LSDA}10/$ $\text{LGPL}0$	O	<p>SDRAM A10/General-purpose line 0</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general-purpose signals when in UPM mode; it drives a value programmed in the UPM array.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general-purpose signals when in UPM mode; it drives a value programmed in the UPM array.		
<b>State Meaning</b>	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general-purpose signals when in UPM mode; it drives a value programmed in the UPM array.					
$\overline{\text{LSDWE}}/$ $\text{LGPL}1$	O	<p>SDRAM write enable/General-purpose line 1</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.		
<b>State Meaning</b>	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.					
$\overline{\text{LOE}}/$ $\overline{\text{LSDRAS}}/$ $\text{LGPL}2$	O	<p>GPCM output enable/SDRAM <math>\overline{\text{RAS}}</math>/General-purpose line 2</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe (<math>\overline{\text{RAS}}</math>). One of six general-purpose lines when in UPM mode; it drives a value programmed in the UPM array.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe ( $\overline{\text{RAS}}$ ). One of six general-purpose lines when in UPM mode; it drives a value programmed in the UPM array.		
<b>State Meaning</b>	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe ( $\overline{\text{RAS}}$ ). One of six general-purpose lines when in UPM mode; it drives a value programmed in the UPM array.					
$\overline{\text{LSDCAS}}/$ $\text{LGPL}3$	O	<p>SDRAM CAS/General-purpose line 3</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—In SDRAM mode, drives the column address strobe (<math>\overline{\text{CAS}}</math>). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—In SDRAM mode, drives the column address strobe ( $\overline{\text{CAS}}$ ). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.		
<b>State Meaning</b>	Asserted/Negated—In SDRAM mode, drives the column address strobe ( $\overline{\text{CAS}}$ ). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.					
$\overline{\text{LGTA}}/$ $\text{LGPL}4/$ $\text{LUPWAIT}/$ $\text{LPBSE}$	I/O	<p>GPCM transfer acknowledge/General-purpose line 4/UPM wait/parity byte select</p> <table border="1"> <tr> <td><b>State Meaning</b></td> <td>Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing <math>\overline{\text{LBS}}[0:3]</math> through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC provides this optional byte-select signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to RMW-parity devices.</td> </tr> </table>	<b>State Meaning</b>	Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing $\overline{\text{LBS}}[0:3]$ through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC provides this optional byte-select signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to RMW-parity devices.		
<b>State Meaning</b>	Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing $\overline{\text{LBS}}[0:3]$ through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC provides this optional byte-select signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to RMW-parity devices.					

Table 14-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description
LGPL5	O	General-purpose line 5
		<b>State Meaning</b> Asserted/Negated—One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
LBCTL	O	Data buffer control. The memory controller activates LBCTL for the local bus when a GPCM- or UPM-controlled bank is accessed. Access to an SDRAM machine-controlled bank does not activate the buffer control. Buffer control is disabled by setting $ORn[BCTLD]$ .
		<b>State Meaning</b> Asserted/Negated—The LBCTL signal normally functions as a write/read control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
LA[27:31]	O	Local bus non-multiplexed address lsbs. All bits driven on LA[27:31] are defined for 8-bit port sizes. For 32-bit port sizes, LA[30:31] are don't cares; for 16-bit port sizes LA31 is a don't care.
		<b>State Meaning</b> Asserted/Negated—Although the LBC shares an address and data bus, up to five lsbs of the RAM address always appear on the dedicated address signals, LA[27:31]. These may be used, unlatched, in place of LAD[27:31] to connect the five lsbs of the address for address phases. For some RAM devices, such as fast-page DRAM, LA[27:31] serve as the column address offset during a burst access.
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in $BRn[PS]$ as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). For a port size of 16 bits, LAD[0:7] connect to the most significant byte lane (at address offset 0), while LAD[8:15] connect to the least-significant byte lane (at address offset 1); LAD[16:31] are unused for 16-bit port sizes. For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.
		<b>State Meaning</b> Asserted/Negated—LAD[0:31] is the shared 32-bit address/data bus through which external RAM devices transfer data and receive addresses.
		<b>Timing</b> Assertion/Negation—During assertion of LALE, LAD[0:31] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:31] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:31] are either driven by write data or are made high impedance by the LBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD[0:31] are again taken into a high-impedance state.
LDP[0:3]	I/O	Local bus data parity. Drives and receives the data parity corresponding with the data phases on LAD[0:31].
		<b>State Meaning</b> Asserted/Negated—During write accesses, a parity bit is generated for each 8 bits of LAD[0:31], such that LDP0 is even/odd parity for LAD[0:7], while LDP3 is even/odd parity for LAD[24:31]. Unused byte lanes for port sizes less than 32 bits have undefined parity.
		<b>Timing</b> Assertion/Negation—Drive and receive the data parity corresponding with the data phases on LAD[0:31]. For read accesses, the parity bits for each byte lane are sampled on LDP[0:3] with the same timing that read data is sampled on LAD[0:31]. LDP[0:3] change impedance in concert with LAD[0:31].
LCKE	O	Local bus clock enable
		<b>State Meaning</b> Asserted/Negated—LCKE is the bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. Asserted during normal SDRAM operation.

Table 14-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LCLK[0:2]	O	Local bus clocks	
		<b>State Meaning</b>	Asserted/Negated—LCLK[0:2] drive an identical bus clock signal for distributed loads. If the LBC PLL is enabled (see LCRR[PBYP], <a href="#">Figure 14-19 on page 14-30</a> ), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:31] and $\overline{\text{LCS}}_n$ ) by a time delay matching the delay of the PLL timing loop set up between LSYNC_OUT and LSYNC_IN.
LSYNC_OUT	O	PLL synchronization out	
		<b>State Meaning</b>	Asserted/Negated—A replica of the bus clock, appearing on LSYNC_OUT, should be propagated through a passive timing loop and returned to LSYNC_IN for achieving correct PLL lock.
		<b>Timing</b>	Assertion/Negation—The time delay of the timing loop should be such that it compensates for the round-trip flight time of LCLK[2] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	PLL synchronization in	
		<b>State Meaning</b>	Asserted/Negated—See description of LSYNC_OUT.
MDVAL	O	Local bus data valid (LBC debug mode only)	
		<b>State Meaning</b>	Asserted/Negated—For a read, MDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:31]. For a write, MDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, MDVAL asserts for each data beat.
		<b>Timing</b>	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, MDVAL asserts when the LBC generates a data transfer acknowledge.
MSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all MSRCID[0:4] signals are driven high unless MSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		<b>State Meaning</b>	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until MDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, MSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC. For example, MSRCID[0:4] is valid only during $\overline{\text{CAS}}$ phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a $\overline{\text{CAS}}$ cycle

### 14.3 Memory Map/Register Definition

[Table 14-3](#) shows the memory mapped registers of the LBC and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in [Table 14-3](#). Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.

- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 14-3. Local Bus Controller Memory Map

Offset	Use	Access	Reset	Section/Page
<b>Local Bus Block Base Address: 0x0_5000</b>				
0x000	BR0—Base register 0	R/W	0x0000_nn01 <sup>1</sup>	14.3.1.1/14-10
0x008	BR1—Base register 1		0x0000_0000	
0x010	BR2—Base register 2			
0x018	BR3—Base register 3			
0x020	BR4—Base register 4			
0x028	BR5—Base register 5			
0x030	BR6—Base register 6			
0x038	BR7—Base register 7			
0x004	OR0—Options register 0	R/W	0x0000_0FF7	14.3.1.2/14-12
0x00C	OR1—Options register 1		0x0000_0000	
0x014	OR2—Options register 2			
0x01C	OR3—Options register 3			
0x024	OR4—Options register 4			
0x02C	OR5—Options register 5			
0x034	OR6—Options register 6			
0x03C	OR7—Options register 7			
0x068	MAR—UPM address register	R/W	0x0000_0000	14.3.1.3/14-17
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	14.3.1.5/14-20
0x088	MDR—UPM data register	R/W	0x0000_0000	14.3.1.6/14-20
0x094	LSDMR—SDRAM mode register	R/W	0x0000_0000	14.3.1.7/14-21
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	14.3.1.8/14-23
0x0A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	14.3.1.9/14-23
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	14.3.1.10/14-24
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	14.3.1.11/14-25

**Table 14-3. Local Bus Controller Memory Map (continued)**

Offset	Use	Access	Reset	Section/Page
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	14.3.1.12/14-26
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	14.3.1.13/14-27
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	14.3.1.14/14-29
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	14.3.1.15/14-29
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	14.3.1.16/14-30

<sup>1</sup> Port size for BR0 is configured from external signals during reset, hence ‘*nn*’ is either 0x08, 0x10, or 0x18. See [Section 4.4.3.4, “Boot ROM Location,”](#) for more information.

### 14.3.1 Register Descriptions

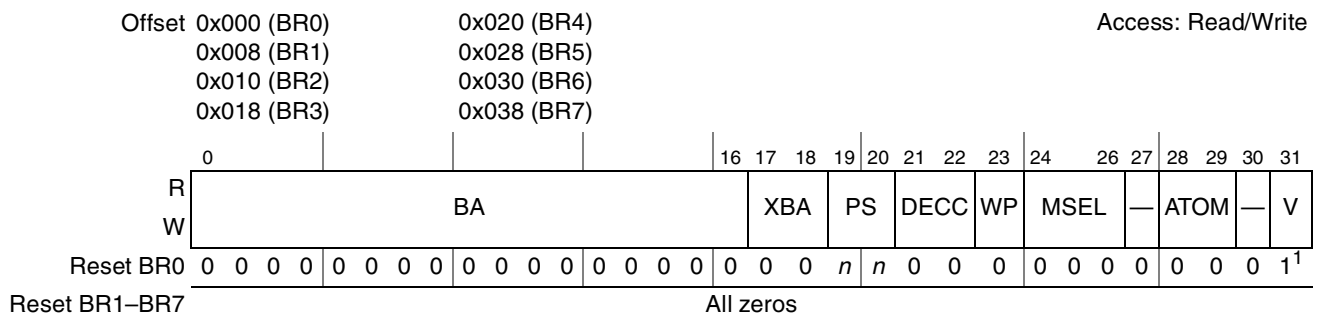
This section provides a detailed description of the LBC configuration, status, and control registers with detailed bit and field descriptions.

Address offsets in the LBC address range that are not defined in [Table 14-3](#) should not be accessed for reading or writing. Similarly, only zero should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

Bits designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

#### 14.3.1.1 Base Registers (BR0–BR7)

The base registers (BR $n$ ), shown in [Figure 14-2](#), contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, BR0[V] is set, BR1[V]–BR7[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location signals.



<sup>1</sup> BR0 has its valid bit set during reset. Thus bank 0 is valid with the port size (PS) configured from external boot ROM configuration signals during reset. (See [Section 4.4.3.4, “Boot ROM Location,”](#) for more information.) All other base registers have all bits cleared to zero during reset.

**Figure 14-2. Base Registers (BR $n$ )**



Table 14-4 describes  $BR_n$  fields.

**Table 14-4.  $BR_n$  Field Descriptions**

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits $OR_n[AM]$ .
17–18	XBA	Extended base address. Extends BA by a further 2 bits such that 19 bits of each base register are compared to the 34-bit transaction address. XBA provides the extra 2 msb's (i.e. {XBA,BA} represents the full base address). Used with the extended address mask bits $OR_n[XAM]$ .
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the boot ROM location signals during reset. For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8-bit 10 16-bit 11 32-bit
21–22	DECC	Specifies the method for data error checking. 00 Data error checking disabled, but normal parity generation 01 Normal parity generation and checking 10 Read-modify-write parity generation and normal parity checking (32-bit port size only) 11 Reserved
23	WP	Write protect 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert $\overline{LCS_n}$ on write cycles to this memory bank. $LTESR[WP]$ is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26	MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (reset value) 001 Reserved 010 Reserved 011 SDRAM 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	—	Reserved
28–29	ATOM	Atomic operation. Writes (reads) to the address space handled by the memory controller bank reserve the selected memory bank for the exclusive use of the accessing device. The reservation is released when the device performs a read (write) operation to this memory controller bank. If a subsequent read (write) request to this memory controller bank is not detected within 256 bus clock cycles of the last write (read), the reservation is released and an atomic error is reported (if enabled). 00 The address space controlled by this bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA) 10 Write-after-read-atomic (WARA) 11 Reserved
30	—	Reserved
31	V	Valid bit. Indicates that the contents of the $BR_n$ and $OR_n$ pair are valid. $\overline{LCS_n}$ does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

### 14.3.1.2 Option Registers (OR0–OR7)

The  $OR_n$  registers define the sizes of memory banks and access attributes. The  $OR_n$  attribute bits support the following three modes of operation as defined by  $BR_n[MSEL]$ .

- GPCM mode
- UPM mode
- SDRAM mode

The  $OR_n$  registers are interpreted differently depending on which of the three machine types is selected for that bank.

#### 14.3.1.2.1 Address Mask

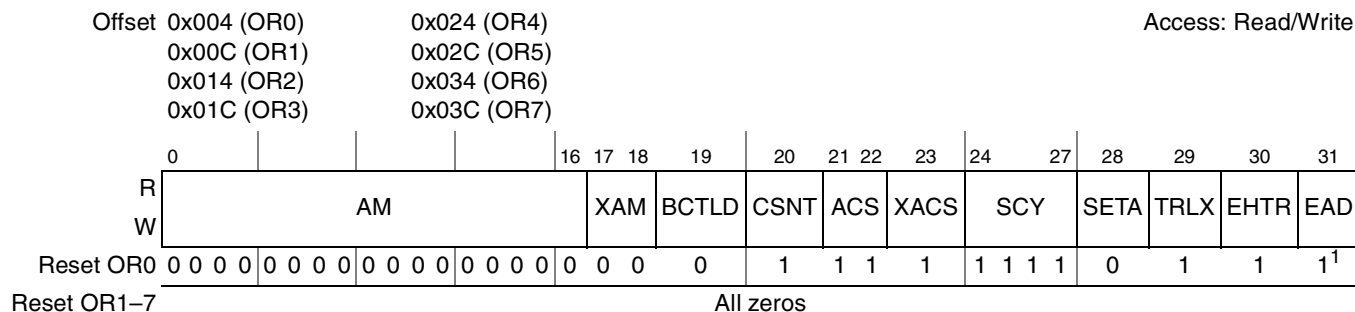
The address mask fields of the option registers ( $OR_n[XAM,AM]$ ) mask up to 19 corresponding  $BR_n[BA,XBA]$  fields. The 15 lsbs of the 34-bit internal address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. Table 14-5 shows memory bank sizes from 32 Kbytes to 4 Gbytes. Note that the use of the  $OR_n[XAM]$  fields provides 34-bits of internal addressing which allows aliasing across 32-bit external banks.

**Table 14-5. Memory Bank Sizes in Relation to Address Mask**

Address Mask	Memory Bank Size	Address Mask	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes	1111_1111_1000_0000_0	8 Mbytes
1000_0000_0000_0000_0	2 Gbytes	1111_1111_1100_0000_0	4 Mbytes
1100_0000_0000_0000_0	1 Gbyte	1111_1111_1110_0000_0	2 Mbytes
1110_0000_0000_0000_0	512 Mbytes	1111_1111_1111_0000_0	1 Mbyte
1111_0000_0000_0000_0	256 Mbytes	1111_1111_1111_1000_0	512 Kbytes
1111_1000_0000_0000_0	128 Mbytes	1111_1111_1111_1100_0	256 Kbytes
1111_1100_0000_0000_0	64 Mbytes	1111_1111_1111_1110_0	128 Kbytes
1111_1110_0000_0000_0	32 Mbytes	1111_1111_1111_1111_0	64 Kbytes
1111_1111_0000_0000_0	16 Mbytes	1111_1111_1111_1111_1	32 Kbytes

### 14.3.1.2.2 Option Registers (OR<sub>n</sub>)—GPCM Mode

Figure 14-3 shows the bit fields for OR<sub>n</sub> when the corresponding BR<sub>n</sub>[MSEL] selects the GPCM machine.



**Figure 14-3. Option Registers (OR<sub>n</sub>) in GPCM Mode**

<sup>1</sup> OR0 has this value set during reset (GPCM is the default control machine for all banks coming out of reset). All other option registers have all bits cleared.

Table 14-6 describes OR<sub>n</sub> fields for GPCM mode.

**Table 14-6. OR<sub>n</sub>—GPCM Field Descriptions**

Bits	Name	Description										
0–16	AM	GPCM address mask. Masks corresponding BR <sub>n</sub> bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.										
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR <sub>n</sub> register, effectively extending the address mask (AM) by 2 bits.										
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.										
20	CSNT	Chip select negation time. Determines when $\overline{LCSn}$ and $\overline{LWE}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{LWE}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{LCSn}$ and $\overline{LWE}$ are negated normally. 1 $\overline{LCSn}$ and $\overline{LWE}$ are negated one quarter of a bus clock cycle earlier.										
21–22	ACS	Address to chip-select setup. Determines the delay of the $\overline{LCSn}$ assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11. <table border="1" style="width: 100%; margin-top: 10px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Meaning</th> </tr> </thead> <tbody> <tr> <td>00</td> <td><math>\overline{LCSn}</math> is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.</td> </tr> <tr> <td>01</td> <td>Reserved.</td> </tr> <tr> <td>10</td> <td><math>\overline{LCSn}</math> is output a quarter bus clock cycle after the address lines.</td> </tr> <tr> <td>11</td> <td><math>\overline{LCSn}</math> is output a half bus clock cycle after the address lines.</td> </tr> </tbody> </table>	Value	Meaning	00	$\overline{LCSn}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.	01	Reserved.	10	$\overline{LCSn}$ is output a quarter bus clock cycle after the address lines.	11	$\overline{LCSn}$ is output a half bus clock cycle after the address lines.
Value	Meaning											
00	$\overline{LCSn}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.											
01	Reserved.											
10	$\overline{LCSn}$ is output a quarter bus clock cycle after the address lines.											
11	$\overline{LCSn}$ is output a half bus clock cycle after the address lines.											

**Table 14-6. OR<sub>n</sub>—GPCM Field Descriptions (continued)**

Bits	Name	Description															
23	XACS	Extra address to chip-select setup. Setting this bit increases the delay of the $\overline{LCSn}$ assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, $OR0[XACS] = 1$ . 0 Address to chip-select setup is determined by $ORx[ACS]$ . 1 Address to chip-select setup is extended (see Table 14-23 and Table 14-24).															
24–27	SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, $OR0[SCY] = 1111$ . 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states															
28	SETA	External address termination 0 Access is terminated internally by the memory controller unless the external device asserts $\overline{LGT\bar{A}}$ earlier to terminate the access. 1 Access is terminated externally by asserting the $\overline{LGT\bar{A}}$ external signal. (Only $\overline{LGT\bar{A}}$ can terminate the access).															
29	TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: <ul style="list-style-type: none"> <li>• Adds an additional cycle between the address and control signals (only if <math>ACS \neq 00</math>)</li> <li>• Doubles the number of wait states specified by SCY, providing up to 30 wait states</li> <li>• Works in conjunction with EHTR to extend hold time on read accesses</li> <li>• <math>\overline{LCSn}</math> (only if <math>ACS \neq 00</math>) and <math>\overline{LWE}</math> signals are negated one cycle earlier during writes.</li> </ul>															
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>TRLX</th> <th>EHTR</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The memory controller generates normal timing. No additional cycles are inserted.</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 idle clock cycle is inserted.</td> </tr> <tr> <td>1</td> <td>0</td> <td>4 idle clock cycles are inserted.</td> </tr> <tr> <td>1</td> <td>1</td> <td>8 idle clock cycles are inserted.</td> </tr> </tbody> </table>	TRLX	EHTR	Meaning	0	0	The memory controller generates normal timing. No additional cycles are inserted.	0	1	1 idle clock cycle is inserted.	1	0	4 idle clock cycles are inserted.	1	1	8 idle clock cycles are inserted.
TRLX	EHTR	Meaning															
0	0	The memory controller generates normal timing. No additional cycles are inserted.															
0	1	1 idle clock cycle is inserted.															
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by $LCRR[EADC]$ ).															

### 14.3.1.2.3 Option Registers (OR<sub>n</sub>)—UPM Mode

Figure 14-4 shows the bit fields for OR<sub>n</sub> when the corresponding BR<sub>n</sub>[MSEL] selects a UPM machine.

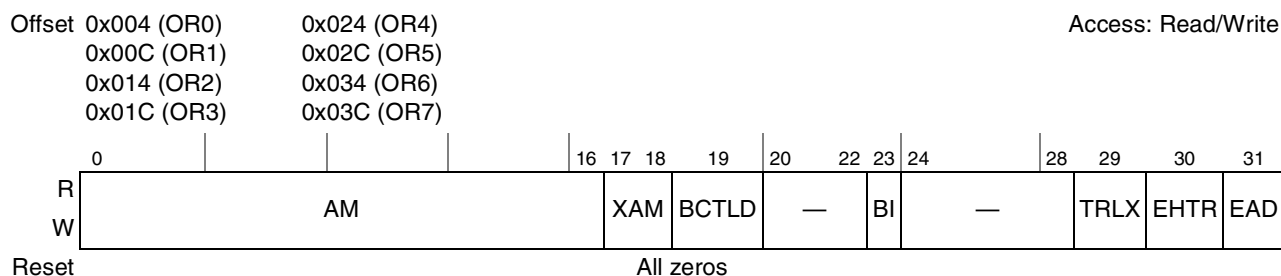


Figure 14-4. Option Registers (OR<sub>n</sub>) in UPM Mode

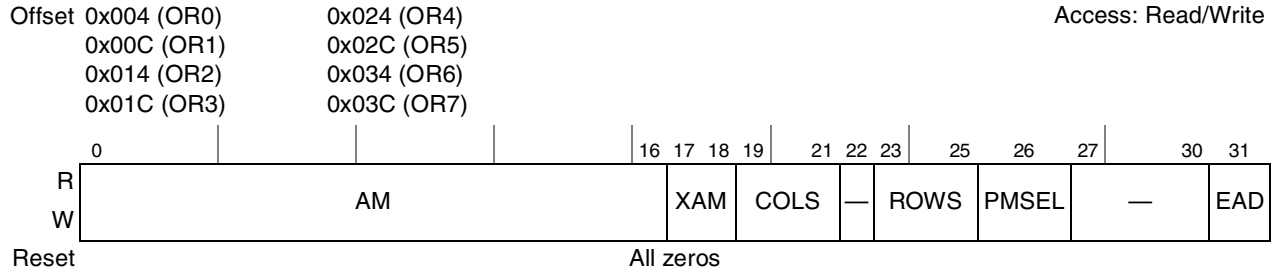
Table 14-7 describes BR<sub>n</sub> fields for UPM mode.

Table 14-7. OR<sub>n</sub>—UPM Field Descriptions

Bits	Name	Description															
0–16	AM	UPM address mask. Masks corresponding BR <sub>n</sub> bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.															
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR <sub>n</sub> register, effectively extending the address mask (AM) by 2 bits.															
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.															
20–22	—	Reserved															
23	BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.															
24–28	—	Reserved															
29	TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.															
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>TRLX</th> <th>EHTR</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The memory controller generates normal timing. No additional cycles are inserted.</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 idle clock cycle is inserted.</td> </tr> <tr> <td>1</td> <td>0</td> <td>4 idle clock cycles are inserted.</td> </tr> <tr> <td>1</td> <td>1</td> <td>8 idle clock cycles are inserted.</td> </tr> </tbody> </table>	TRLX	EHTR	Meaning	0	0	The memory controller generates normal timing. No additional cycles are inserted.	0	1	1 idle clock cycle is inserted.	1	0	4 idle clock cycles are inserted.	1	1	8 idle clock cycles are inserted.
TRLX	EHTR	Meaning															
0	0	The memory controller generates normal timing. No additional cycles are inserted.															
0	1	1 idle clock cycle is inserted.															
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).															

### 14.3.1.2.4 Option Registers (OR<sub>n</sub>)—SDRAM Mode

Figure 14-5 shows the bit fields for OR<sub>n</sub> when the corresponding BR<sub>n</sub>[MSEL] selects the SDRAM machine.



**Figure 14-5. Option Registers (OR<sub>n</sub>) in SDRAM Mode**

Table 14-8 describes BR<sub>n</sub> fields for SDRAM mode.

**Table 14-8. OR<sub>n</sub>—SDRAM Field Descriptions**

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BR <sub>n</sub> bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	XAM	Extended address mask. Masks the corresponding BR <sub>n</sub> [XBA] bits, effectively extending the address mask (AM) by 2 bits.
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7                                  100 11 001 8                                  101 12 010 9                                  110 13 011 10                                 111 14
22	—	Reserved
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9                                    100 13 001 10                                  101 14 010 11                                 110 15 011 12                                 111 Reserved
26	PMSEL	Page mode select. Selects page mode for the SDRAM connected to the memory controller bank. 0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.
27–30	—	Reserved
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

### 14.3.1.3 UPM Memory Address Register (MAR)

Figure 14-6 shows the fields of the UPM memory address register (MAR).

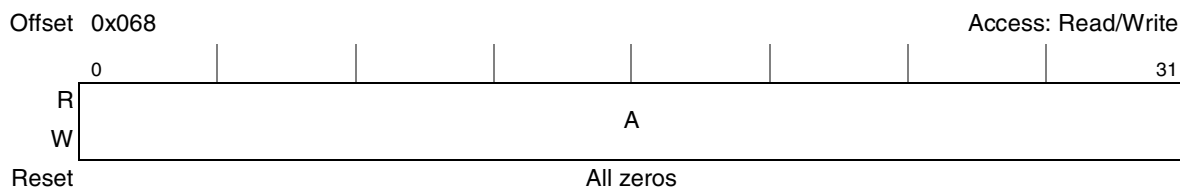


Figure 14-6. UPM Memory Address Register (MAR)

Table 14-9 describes the MAR fields.

Table 14-9. MAR Field Descriptions

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

### 14.3.1.4 UPM Mode Registers (MxMR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 14-7, contain the configuration for the three UPMs.

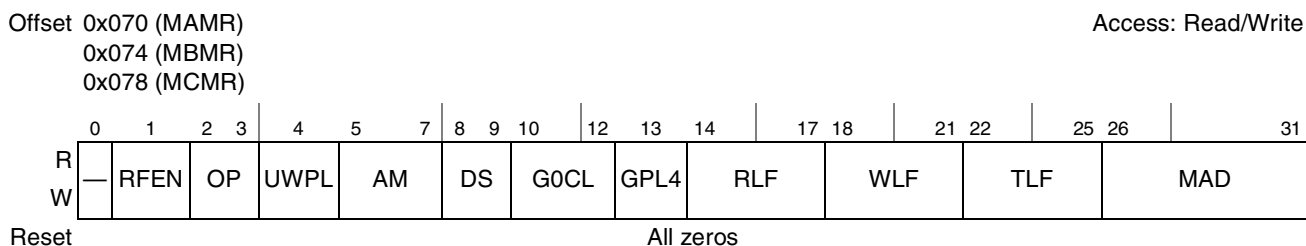


Figure 14-7. UPM Mode Registers (MxMR)

Table 14-10 describes UPM mode fields.

Table 14-10. MxMR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the UPM needs refresh services. This bit must be set for UPMA (refresh executor) if refresh services are required on any UPM assigned chip selects. If MAMR[RFEN] = 0, no refresh services can be provided, even if UPMB and/or UPMC have their RFEN bit set. 0 Refresh services are not required 1 Refresh services are required

Table 14-10. MxMR Field Descriptions (continued)

Bits	Name	Description																																																																																	
2–3	OP	<p>Command opcode. Determines the command executed by the UPM<math>n</math> when a memory access hits a UPM assigned bank. See <a href="#">Section 14.4.4.2, “Programming the UPMs,”</a> for important programming considerations.</p> <p>00 Normal operation            01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented.            10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented.            11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.</p>																																																																																	
4	UWPL	<p>LUPWAIT polarity active low. Sets the polarity of the LUPWAIT signal when in UPM mode.</p> <p>0 LUPWAIT is active high.            1 LUPWAIT is active low.</p>																																																																																	
5–7	AM	<p>Address multiplex size. Determines how the address of the current memory cycle can be output on the address signals. This field is needed when interfacing with devices requiring row and column addresses multiplexed on the same signals.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Value</th> <th>LA0–LA15</th> <th>LA16</th> <th>LA17</th> <th>LA18</th> <th>LA19–LA28</th> <th>LA29</th> <th>LA30</th> <th>LA31</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>0</td> <td>A8</td> <td>A9</td> <td>A10</td> <td>A11–A20</td> <td>A21</td> <td>A22</td> <td>A23</td> </tr> <tr> <td>001</td> <td>0</td> <td>A7</td> <td>A8</td> <td>A9</td> <td>A10–A19</td> <td>A20</td> <td>A21</td> <td>A22</td> </tr> <tr> <td>010</td> <td>0</td> <td>A6</td> <td>A7</td> <td>A8</td> <td>A9–A18</td> <td>A19</td> <td>A20</td> <td>A21</td> </tr> <tr> <td>011</td> <td>0</td> <td>A5</td> <td>A6</td> <td>A7</td> <td>A8–A17</td> <td>A18</td> <td>A19</td> <td>A20</td> </tr> <tr> <td>100</td> <td>0</td> <td>A4</td> <td>A5</td> <td>A6</td> <td>A7–A16</td> <td>A17</td> <td>A18</td> <td>A19</td> </tr> <tr> <td>101</td> <td>0</td> <td>A3</td> <td>A4</td> <td>A5</td> <td>A6–A15</td> <td>A16</td> <td>A17</td> <td>A18</td> </tr> <tr> <td>110</td> <td colspan="8" style="text-align: center;">Reserved</td> </tr> <tr> <td>111</td> <td colspan="8" style="text-align: center;">Reserved</td> </tr> </tbody> </table>	Value	LA0–LA15	LA16	LA17	LA18	LA19–LA28	LA29	LA30	LA31	000	0	A8	A9	A10	A11–A20	A21	A22	A23	001	0	A7	A8	A9	A10–A19	A20	A21	A22	010	0	A6	A7	A8	A9–A18	A19	A20	A21	011	0	A5	A6	A7	A8–A17	A18	A19	A20	100	0	A4	A5	A6	A7–A16	A17	A18	A19	101	0	A3	A4	A5	A6–A15	A16	A17	A18	110	Reserved								111	Reserved							
Value	LA0–LA15	LA16	LA17	LA18	LA19–LA28	LA29	LA30	LA31																																																																											
000	0	A8	A9	A10	A11–A20	A21	A22	A23																																																																											
001	0	A7	A8	A9	A10–A19	A20	A21	A22																																																																											
010	0	A6	A7	A8	A9–A18	A19	A20	A21																																																																											
011	0	A5	A6	A7	A8–A17	A18	A19	A20																																																																											
100	0	A4	A5	A6	A7–A16	A17	A18	A19																																																																											
101	0	A3	A4	A5	A6–A15	A16	A17	A18																																																																											
110	Reserved																																																																																		
111	Reserved																																																																																		
8–9	DS	<p>Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by UPM<math>n</math>. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPM<math>n</math> allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPM<math>n</math> is also allowed. To avoid conflicts between successive accesses to different banks, the minimum pattern in the RAM array for a request serviced, should not be shorter than the period established by DS.</p> <p>00 1-bus clock cycle disable period            01 2-bus clock cycle disable period            10 3-bus clock cycle disable period            11 4-bus clock cycle disable period</p>																																																																																	

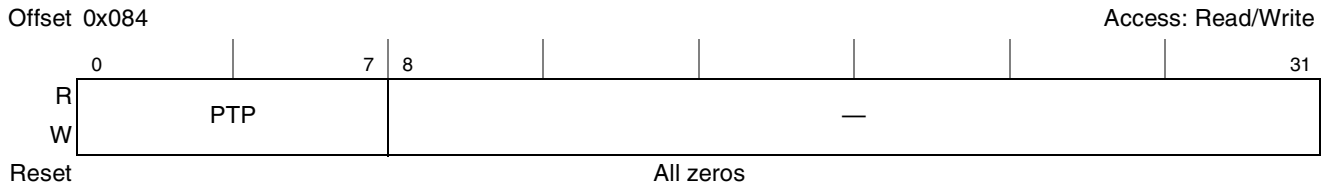


Table 14-10. MxMR Field Descriptions (continued)

Bits	Name	Description														
10–12	G0CL	General line 0 control. Determines which logical address line can be output to the LGPL0 signal when the UPM $n$ is selected to control the memory access. 000 A12 001 A11 010 A10 011 A9 100 A8 101 A7 110 A6 111 A5														
13	GPL4	LGPL4 output line disable. Determines how the LGPL4/LUPWAIT signal is controlled by the corresponding bits in the UPM $n$ array. See <a href="#">Table 14-28</a> . <table border="1" data-bbox="505 667 1300 877"> <thead> <tr> <th rowspan="2">Value</th> <th rowspan="2">LGPL4/LUPWAIT Signal Function</th> <th colspan="2">Interpretation of UPM Word Bits</th> </tr> <tr> <th>G4T1/DLT3</th> <th>G4T3/WAEN</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>LGPL4 (output)</td> <td>G4T1</td> <td>G4T3</td> </tr> <tr> <td>1</td> <td>LUPWAIT (input)</td> <td>DLT3</td> <td>WAEN</td> </tr> </tbody> </table>	Value	LGPL4/LUPWAIT Signal Function	Interpretation of UPM Word Bits		G4T1/DLT3	G4T3/WAEN	0	LGPL4 (output)	G4T1	G4T3	1	LUPWAIT (input)	DLT3	WAEN
Value	LGPL4/LUPWAIT Signal Function	Interpretation of UPM Word Bits														
		G4T1/DLT3	G4T3/WAEN													
0	LGPL4 (output)	G4T1	G4T3													
1	LUPWAIT (input)	DLT3	WAEN													
14–17	RLF	Read loop field. Determines the number of times a loop defined in the UPM $n$ will be executed for a burst- or single-beat read pattern or when MxMR[OP] = 11 (RUN command) 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15														
18–21	WLF	Write loop field. Determines the number of times a loop defined in the UPM $n$ will be executed for a burst- or single-beat write pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15														
22–25	TLF	Refresh loop field. Determines the number of times a loop defined in the UPM $n$ will be executed for a refresh service pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15														
26–31	MAD	Machine address. RAM address pointer for the command executed. This field is incremented by 1 each time the UPM is accessed, and the OP field is set to WRITE or READ. Address range is 64 words per UPM $n$ .														

### 14.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in [Figure 14-8](#), is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.



**Figure 14-8. Memory Refresh Timer Prescaler Register (MRTPR)**

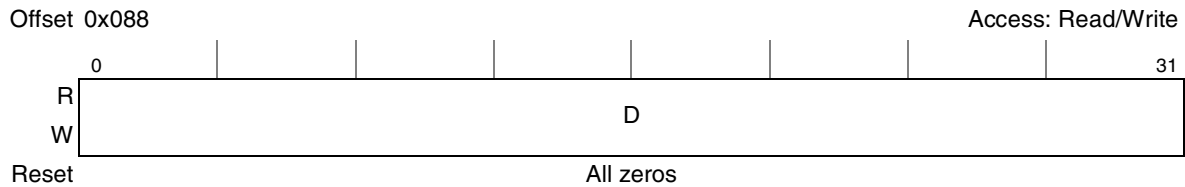
[Table 14-11](#) describes MRTPR fields.

**Table 14-11. MRTPR Field Descriptions**

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

### 14.3.1.6 UPM Data Register (MDR)

The memory data register (MDR), shown in [Figure 14-9](#), contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.



**Figure 14-9. UPM Data Register (MDR)**

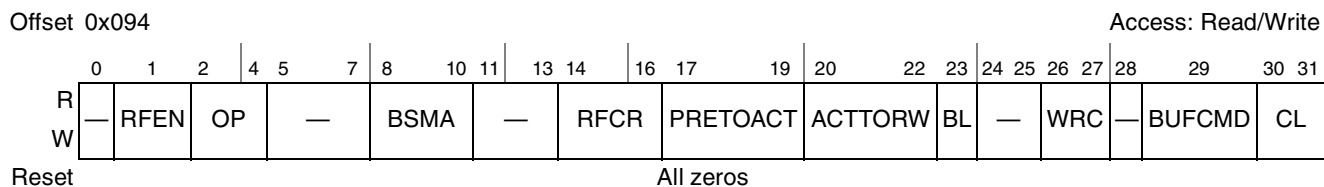
[Table 14-12](#) describes MDR[D].

**Table 14-12. MDR Field Descriptions**

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM (MxMR[OP] = 01 or MxMR[OP] = 10).

### 14.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR), shown in [Figure 14-10](#), is used to configure operations pertaining to SDRAM.



**Figure 14-10. SDRAM Machine Mode Register (LSDMR)**

[Table 14-12](#) describes LSDMR fields.

**Table 14-13. LSDMR Field Descriptions**

Bits	Name	Description																											
0	—	Reserved																											
1	RFEN	Refresh enable. Indicates that the SDRAM requires refresh services. 0 Refresh services are not required. 1 Refresh services are required.																											
2–4	OP	SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th>Value</th> <th>Meaning</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>Normal operation</td> <td>Normal operation</td> </tr> <tr> <td>001</td> <td>Auto refresh</td> <td>Initialization</td> </tr> <tr> <td>010</td> <td>Self refresh</td> <td>Power-down mode or debug</td> </tr> <tr> <td>011</td> <td>Mode Register write</td> <td>Initialization</td> </tr> <tr> <td>100</td> <td>Precharge bank</td> <td>Debug</td> </tr> <tr> <td>101</td> <td>Precharge all banks</td> <td>Initialization</td> </tr> <tr> <td>110</td> <td>Activate bank</td> <td>Debug</td> </tr> <tr> <td>111</td> <td>Read/write without valid data transfer</td> <td>Debug</td> </tr> </tbody> </table>	Value	Meaning	Use	000	Normal operation	Normal operation	001	Auto refresh	Initialization	010	Self refresh	Power-down mode or debug	011	Mode Register write	Initialization	100	Precharge bank	Debug	101	Precharge all banks	Initialization	110	Activate bank	Debug	111	Read/write without valid data transfer	Debug
Value	Meaning	Use																											
000	Normal operation	Normal operation																											
001	Auto refresh	Initialization																											
010	Self refresh	Power-down mode or debug																											
011	Mode Register write	Initialization																											
100	Precharge bank	Debug																											
101	Precharge all banks	Initialization																											
110	Activate bank	Debug																											
111	Read/write without valid data transfer	Debug																											
5–7	—	Reserved																											
8–10	BSMA	Bank select multiplexed address line. Selects which address signals serve as the 2-bit bank-select address for SDRAM. Note that only 4-bank SDRAMs are supported. 000 LA12:LA13                      100 LA16:LA17 001 LA13:LA14                      101 LA17:LA18 010 LA14:LA15                      110 LA18:LA19 011 LA15:LA16                      111 LA19:LA20																											
11–13	—	Reserved																											

Table 14-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
14–16	RFCR	Refresh recovery. Sets the refresh recovery interval in bus clock cycles. Defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command. 000 Reserved                      100 6 clocks 001 3 clocks                        101 7 clocks 010 4 clocks                        110 8 clocks 011 5 clocks                        111 16 clocks
17–19	PRETOACT	Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states). 000 8                                  100 4 001 1                                  101 5 010 2                                  110 6 011 3                                  111 7
20–22	ACTTORW	Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). 000 8                                  100 4 001 Reserved                        101 5 010 2                                  110 6 011 3                                  111 7
23	BL	Sets the burst length for SDRAM accesses. 0 SDRAM burst length is 4. Use this value if the device port size is 16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
24–25	—	Reserved
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data is written to the SDRAM. 00 4 01 Reserved 10 2 11 3
28	—	Reserved
29	BUFCMD	Control line assertion timing. If external buffers are placed on the control lines going to both the SDRAM and address lines, setting BUFCMD causes all SDRAM control lines except $\overline{LCSn}$ , $\overline{LCKE}$ , $\overline{LALE}$ , and $\overline{LSDDQM}[0:3]$ to be asserted for $\text{LCRR}[\text{BUFCMDC}]$ cycles, instead of one. 0 Normal timing for the control lines 1 All control lines except $\overline{LCSn}$ are asserted for the number of cycles specified by $\text{LCRR}[\text{BUFCMDC}]$ .
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. 00 Extended CAS latency. According to $\text{LCRR}[\text{ECL}]$ . See <a href="#">Table 14-22</a> . 01 1 10 2 11 3

### 14.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 14-11, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled ( $MxMR[RFEN] = 1$ ). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

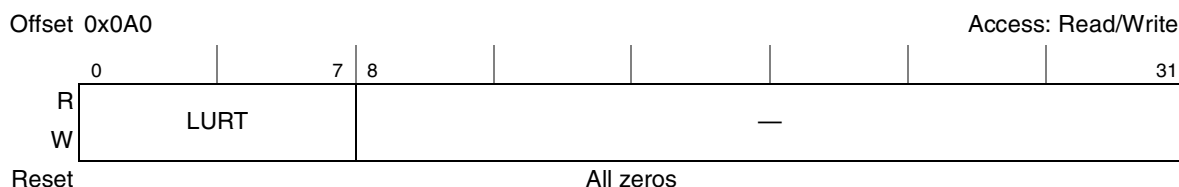


Figure 14-11. UPM Refresh Timer (LURT)

Table 14-14 describes LURT fields.

Table 14-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LURT}}{\left(\frac{F_{\text{systemclock}}}{\text{MRTPR}[PTP]}\right)}$ <p><b>Example:</b> For a 266-MHz system clock and a required service rate of 15.6 <math>\mu\text{s}</math>, given <math>\text{MRTPR}[PTP] = 32</math>, the LURT value should be 128 decimal. <math>128 / (266 \text{ MHz} / 32) = 15.4 \mu\text{s}</math>, which is less than the required service period of 15.6 <math>\mu\text{s}</math>. Note that the reset value (0x00) sets the maximum period to <math>256 \times \text{MRTPR}[PTP]</math> system clock cycles.</p>
8–31	—	Reserved

### 14.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT), shown in Figure 14-12, generates a refresh request for all valid banks that selected a SDRAM machine and are refresh-enabled ( $\text{LSDMR}[RFEN] = 1$ ). Each time the timer expires, all qualifying banks generate a bank staggering auto-refresh request using the SDRAM machine.



Figure 14-12. LSRT SDRAM Refresh Timer (LSRT)

Table 14-15 describes LSRT fields.

**Table 14-15. LSRT Field Descriptions**

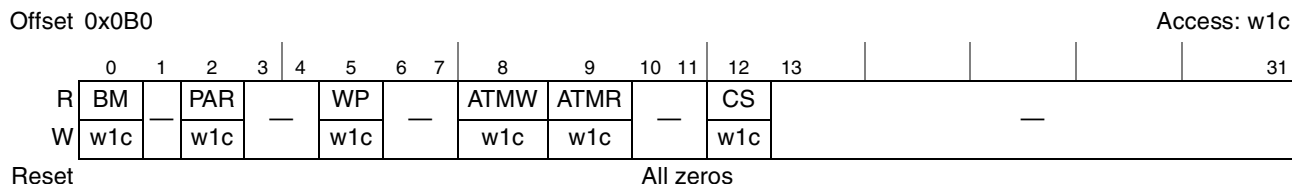
Bits	Name	Description
0–7	LSRT	<p>SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation:</p> $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{F_{\text{systemclock}}}{\text{MRTPR}[\text{PTP}]}\right)}$ <p><b>Example:</b> For a 266-MHz system clock and a required service rate of 15.6 <math>\mu\text{s}</math>, given PTP = 32, the LSRT value should be 128 decimal. <math>128/(266 \text{ MHz}/32) = 15.4 \mu\text{s}</math>, which is less than the required service period of 15.6 <math>\mu\text{s}</math>. Note that the reset value, 0x00, sets the maximum period to <math>256 \times \text{MRTPR}[\text{PTP}]</math> system clock cycles.</p>
8–31	—	Reserved

### 14.3.1.10 Transfer Error Status Register (LTESR)

The LBC has the following registers for error management:

- The transfer error status register (LTESR) indicates the cause of an error.
- The transfer error check disable register (LTEDR) is used to enable (and disable) error checking.
- The transfer error check interrupt register (LTEIR) enables reporting of errors through an interrupt.
- The transfer error attributes register (LTEATR) captures source attributes of an error.
- The transfer error address register (LTEAR) captures the address of a transaction that caused an error.

LTESR, shown in Figure 14-13, is a write-one-to-clear register. Reading LTESR occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. For example, to clear only the write protect error bit (LTESR[WP]) without affecting other LTESR bits, 0x0400\_0000 should be written to the register. Note that LTEATR[V] bit has to be cleared to register subsequent errors in LTESR.



**Figure 14-13. Transfer Error Status Register (LTESR)**

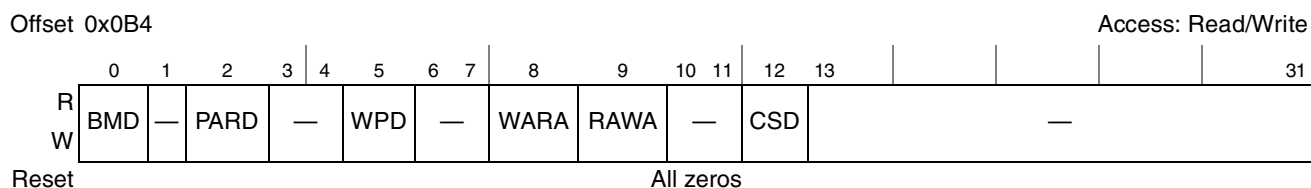
Table 14-16 describes LTESR fields.

**Table 14-16. LTESR Field Descriptions**

Bits	Name	Description
0	BM	Bus monitor time-out 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within $LBCR[BMT] \times 8$ bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	Parity 0 No local bus parity error 1 Local bus parity error. LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BNK] indicates which memory controller bank was accessed.
3–4	—	Reserved
5	WP	Write protect error 0 No write protect error occurred. 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur (as the cycle is not automatically terminated).
6–7	—	Reserved
8	ATMW	Atomic error write 0 No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.
9	ATMR	Atomic error read 0 No atomic read error occurred. 1 The subsequent read (RAWA) to a memory bank did not occur within 256 bus clock cycles.
10–11	—	Reserved
12	CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the LBC that did not hit any memory bank.
13–31	—	Reserved

### 14.3.1.11 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 14-14, is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) through the interrupt mechanism.



**Figure 14-14. Transfer Error Check Disable Register (LTEDR)**

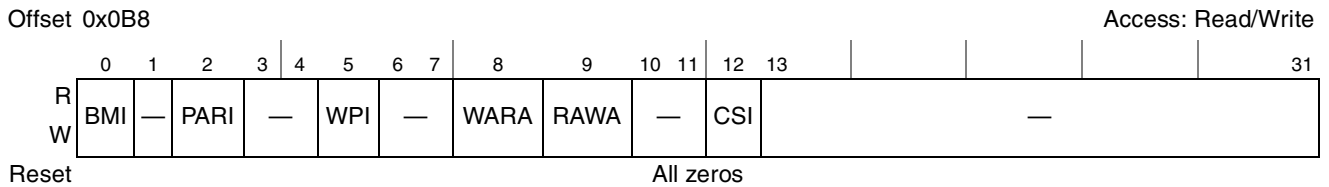
Table 14-17 describes LTEDR fields.

**Table 14-17. LTEDR Field Descriptions**

Bits	Name	Description
0	BMD	Bus monitor disable 0 Bus monitor is enabled 1 Bus monitor is disabled
1	—	Reserved
2	PARD	Parity error checking disabled. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, PARD must be cleared and LTEIR[PARI] must be set to ensure that an interrupt is generated. For more information, see <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”</a> 0 Parity error checking is enabled. 1 Parity error checking is disabled.
3–4	—	Reserved
5	WPD	Write protect error checking disable 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write-after-read atomic (WARA) error checking disable 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read-after-write atomic (RAWA) error checking disable 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved
12	CSD	Chip select error checking disable 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

### 14.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

The transfer error interrupt enable register (LTEIR), shown in Figure 14-15, is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.



**Figure 14-15. Transfer Error Interrupt Enable Register (LTEIR)**



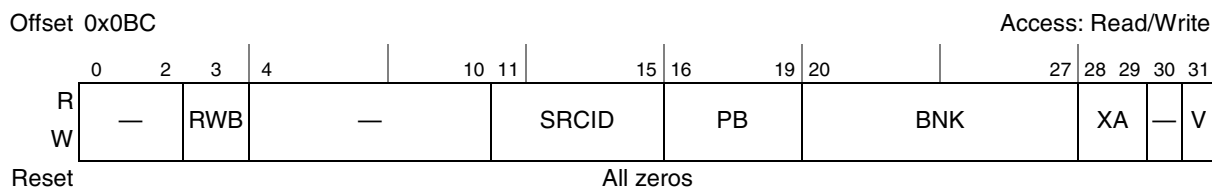
Table 14-18 describes LTEIR fields.

**Table 14-18. LTEIR Field Descriptions**

Bits	Name	Description
0	BMI	Bus monitor error interrupt enable 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1	—	Reserved
2	PARI	Parity error interrupt enable. Note that uncorrectable read errors may cause the assertion of <i>core_fault_in</i> , which causes the core to generate a machine check interrupt, unless it is disabled (by clearing <i>HID1[RFXE]</i> ). If <i>RFXE</i> is zero and this error occurs, <i>LTEDR[PARD]</i> must be cleared and <i>PARI</i> must be set to ensure that an interrupt is generated. For more information, see <a href="#">Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”</a> 0 Parity error reporting is disabled. 1 Parity error reporting is enabled.
3–4	—	Reserved
5	WPI	Write protect error interrupt enable 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–7	—	Reserved
8	WARA	Write-after-read atomic (WARA) error interrupt enable 0 WARA error reporting is disabled. 1 WARA error reporting is enabled.
9	RAWA	Read-after-write atomic (RAWA) error interrupt enable 0 RAWA error reporting is disabled. 1 RAWA error reporting is enabled.
10–11	—	Reserved
12	CSI	Chip select error interrupt enable 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

### 14.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 14-16 shows the LTEATR. After *LTEATR[V]* has been set, software must clear this bit to allow LBC error registers to update following any subsequent errors.



**Figure 14-16. Transfer Error Attributes Register (LTEATR)**

Table 14-19 describes LTEATR fields.

**Table 14-19. LTEATR Field Descriptions**

Bits	Name	Description																														
0–2	—	Reserved																														
3	RWB	Transaction type for the error 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.																														
4–10	—	Reserved																														
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the LBC. <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">00000 PCI1/PCI-X interface</td> <td style="width: 50%;">10010 Reserved</td> </tr> <tr> <td>00001 PCI2 interface</td> <td>10011 Reserved</td> </tr> <tr> <td>00010 PCI Express</td> <td>10100 Reserved</td> </tr> <tr> <td>00011 Reserved</td> <td>10101 DMA</td> </tr> <tr> <td>00100 Local Bus</td> <td>10110 Reserved</td> </tr> <tr> <td>00101–00111 Reserved</td> <td>10111 SAP</td> </tr> <tr> <td>01000 Configuration space</td> <td>11000 eTSEC 1</td> </tr> <tr> <td>01001 Reserved</td> <td>11001 eTSEC 2</td> </tr> <tr> <td>01010 Boot sequencer</td> <td>11010 eTSEC 3</td> </tr> <tr> <td>01011 Reserved</td> <td>11011 eTSEC 4</td> </tr> <tr> <td>01100 Serial RapidIO</td> <td>11100 RapidIO Message Unit</td> </tr> <tr> <td>01101–01110 Reserved</td> <td>11101 RapidIO Doorbell Unit</td> </tr> <tr> <td>01111 DDR controller</td> <td>11110 RapidIO Port-write Unit</td> </tr> <tr> <td>10000 Processor (instruction)</td> <td>11111 Reserved</td> </tr> <tr> <td>10001 Processor (data)</td> <td></td> </tr> </table>	00000 PCI1/PCI-X interface	10010 Reserved	00001 PCI2 interface	10011 Reserved	00010 PCI Express	10100 Reserved	00011 Reserved	10101 DMA	00100 Local Bus	10110 Reserved	00101–00111 Reserved	10111 SAP	01000 Configuration space	11000 eTSEC 1	01001 Reserved	11001 eTSEC 2	01010 Boot sequencer	11010 eTSEC 3	01011 Reserved	11011 eTSEC 4	01100 Serial RapidIO	11100 RapidIO Message Unit	01101–01110 Reserved	11101 RapidIO Doorbell Unit	01111 DDR controller	11110 RapidIO Port-write Unit	10000 Processor (instruction)	11111 Reserved	10001 Processor (data)	
00000 PCI1/PCI-X interface	10010 Reserved																															
00001 PCI2 interface	10011 Reserved																															
00010 PCI Express	10100 Reserved																															
00011 Reserved	10101 DMA																															
00100 Local Bus	10110 Reserved																															
00101–00111 Reserved	10111 SAP																															
01000 Configuration space	11000 eTSEC 1																															
01001 Reserved	11001 eTSEC 2																															
01010 Boot sequencer	11010 eTSEC 3																															
01011 Reserved	11011 eTSEC 4																															
01100 Serial RapidIO	11100 RapidIO Message Unit																															
01101–01110 Reserved	11101 RapidIO Doorbell Unit																															
01111 DDR controller	11110 RapidIO Port-write Unit																															
10000 Processor (instruction)	11111 Reserved																															
10001 Processor (data)																																
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the LBC. <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">00000 PCI interface</td> <td style="width: 50%;">10010 Reserved</td> </tr> <tr> <td>00001 PCI Express interface 2</td> <td>10011 Reserved</td> </tr> <tr> <td>00010 PCI Express interface 1</td> <td>10100 Reserved</td> </tr> <tr> <td>00011 PCI Express interface 3</td> <td>10101 DMA</td> </tr> <tr> <td>00100 Local Bus</td> <td>10110 Reserved</td> </tr> <tr> <td>00101–00111 Reserved</td> <td>10111 SAP</td> </tr> <tr> <td>01000 Configuration space</td> <td>11000 eTSEC 1</td> </tr> <tr> <td>01001 Reserved</td> <td>11001 Reserved</td> </tr> <tr> <td>01010 Boot sequencer</td> <td>11010 eTSEC 3</td> </tr> <tr> <td>01011–01110 Reserved</td> <td>11011 Reserved</td> </tr> <tr> <td>01111 DDR controller</td> <td>11100 Reserved</td> </tr> <tr> <td>10000 Processor (instruction)</td> <td>11101 Reserved</td> </tr> <tr> <td>10001 Processor (data)</td> <td>11110 Reserved</td> </tr> <tr> <td></td> <td>11111 Reserved</td> </tr> </table>	00000 PCI interface	10010 Reserved	00001 PCI Express interface 2	10011 Reserved	00010 PCI Express interface 1	10100 Reserved	00011 PCI Express interface 3	10101 DMA	00100 Local Bus	10110 Reserved	00101–00111 Reserved	10111 SAP	01000 Configuration space	11000 eTSEC 1	01001 Reserved	11001 Reserved	01010 Boot sequencer	11010 eTSEC 3	01011–01110 Reserved	11011 Reserved	01111 DDR controller	11100 Reserved	10000 Processor (instruction)	11101 Reserved	10001 Processor (data)	11110 Reserved		11111 Reserved		
00000 PCI interface	10010 Reserved																															
00001 PCI Express interface 2	10011 Reserved																															
00010 PCI Express interface 1	10100 Reserved																															
00011 PCI Express interface 3	10101 DMA																															
00100 Local Bus	10110 Reserved																															
00101–00111 Reserved	10111 SAP																															
01000 Configuration space	11000 eTSEC 1																															
01001 Reserved	11001 Reserved																															
01010 Boot sequencer	11010 eTSEC 3																															
01011–01110 Reserved	11011 Reserved																															
01111 DDR controller	11100 Reserved																															
10000 Processor (instruction)	11101 Reserved																															
10001 Processor (data)	11110 Reserved																															
	11111 Reserved																															
16–19	PB	Parity error on byte. There are four parity error status bits, one per byte lane. A bit is set for the byte that had a parity error (bit 16 represents byte 0, the most significant byte lane).																														
20–27	BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error. Note that BNK is invalid if the error was not caused by parity checks.																														
28–29	XA	Extended address for the error. These bits capture the two msbs of the 34-bit address of the transaction resulting in an error.																														

Table 14-19. LTEATR Field Descriptions (continued)

Bits	Name	Description
30	—	Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid 0 Captured error attributes and address are not valid 1 Captured error attributes and address are valid

#### 14.3.1.14 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in [Figure 14-17](#).

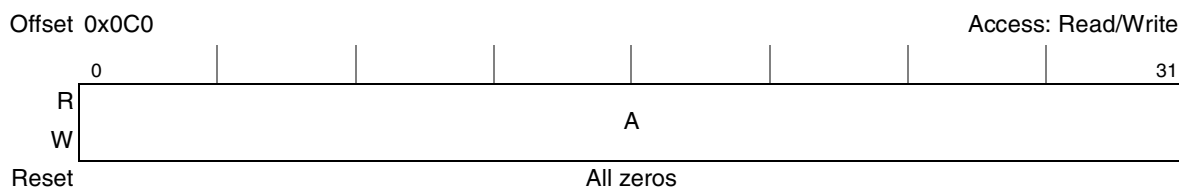


Figure 14-17. Transfer Error Address Register (LTEAR)

[Table 14-20](#) describes LTEAR fields.

Table 14-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Holds the 32 lsbs of the 34-bit address of the transaction resulting in an error.

#### 14.3.1.15 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in [Figure 14-18](#).

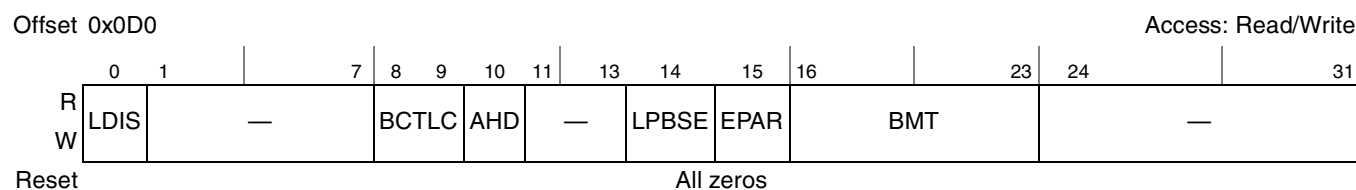


Figure 14-18. Local Bus Configuration Register

[Table 14-21](#) describes LBCR fields.

Table 14-21. LBCR Field Descriptions

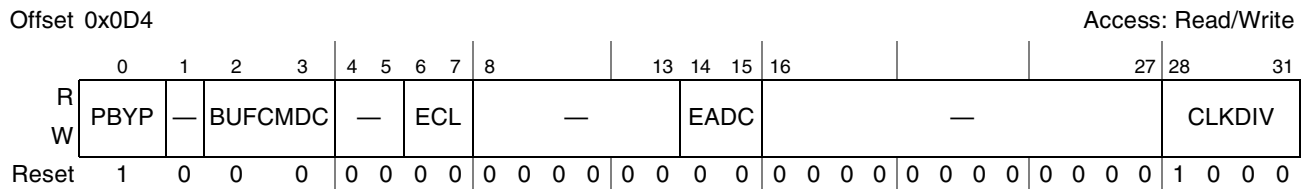
Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved

**Table 14-21. LBCR Field Descriptions (continued)**

Bits	Name	Description
8–9	BCTLC	Defines the use of LBCTL. 00 LBCTL is used as $\overline{W/R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ for GPCM accesses only. 11 Reserved.
10	AHD	Address hold disable. Removes part of the hold time for LAD with respect to LALE in order to lengthen the LALE pulse 0 During address phases on the local bus, the LALE signal negates two platform clock periods prior to the address being invalidated. At 666 MHz, this provides 3 ns of additional address hold time at the external address latch. 1 During address phases on the local bus, the LALE signal negates one platform clock period prior to the address being invalidated. This halves the address hold time, but extends the latch enable duration. This may be necessary for very high frequency designs.
11–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA/LGPL4/LUPWAIT/LPBSE}$ signal. 0 Parity byte select is disabled. $\overline{LGTA/LGPL4/LUPWAIT/LPBSE}$ signal is available for memory control as LGPL4 (output) or $\overline{LGTA/LUPWAIT}$ (input). 1 Parity byte select is enabled. $\overline{LGTA/LGPL4/LUPWAIT/LPBSE}$ signal is dedicated as the parity byte select output, and $\overline{LGTA/LUPWAIT}$ is disabled.
15	EPAR	Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity 1 Even parity
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 2048 bus clock cycles. For non-zero values of BMT, the number of LCLK clock cycles to count down before a time-out error is generated is given by: bus cycles = BMT x 8. Apart from BMT = 0x00, the minimum value of BMT is 5, corresponding with 40 bus cycles. Shorter time-outs may result in spurious errors during SDRAM operation.
24–31	—	Reserved

### 14.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register sets the system (CCB) clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.



**Figure 14-19. Clock Ratio Register (LCRR)**

Table 14-22 describes LCRR fields.

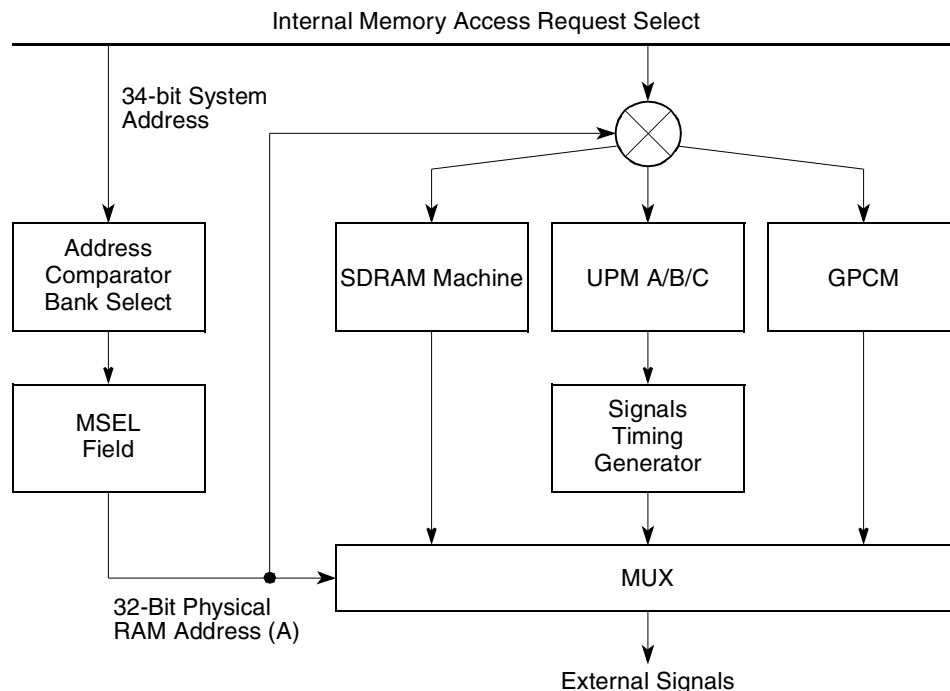
**Table 14-22. LCRR Field Descriptions**

Bits	Name	Description
0	PBYP	PLL bypass. This bit should be set when using low bus clock frequencies if the PLL is unable to lock. When in PLL bypass mode, incoming data is captured in the middle of the bus clock cycle. It is recommended that PLL bypass mode be used at frequencies of 83 MHz or less. 0 The PLL is enabled. 1 The PLL is bypassed.
1	—	Reserved
2–3	BUFCMDC	Additional delay cycles for SDRAM control signals. Defines the number of cycles to be added for each SDRAM command when LSDMR[BUFCMDC] = 1. 00 4 01 1 10 2 11 3
4–5	—	Reserved
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved
14–15	EADC	Additional external address delay cycles. Defines the number of additional cycles for the assertion of LALE. Note that LALE negates prior to the end of the final local bus clock, as controlled by LBCR[AHD]. 00 4 01 1 10 2 11 3
16–27	—	Reserved
28–31	CLKDIV	System (CCB) clock divider. Sets the frequency ratio between the system (CCB) clock and the memory bus clock. Only the values shown in the table below are allowed. <b>Note:</b> It is critical that no transactions are being executed via the local bus while CLKDIV is being modified. As such, prior to modification, the user must ensure that code is not executing out of the local bus. Once LCRR[CLKDIV] is written, the register should be read, and then an isync should be executed. 0000–0001 Reserved 0010 4 0011 Reserved 0100 8 0101–0111 Reserved 1000 16 1001–1111 Reserved

## 14.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to SDRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal PLL for bus clock generation ensures improved data set-up margins for board designs.
- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- The UPM supports refresh timers, address multiplexing of the external bus and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.



**Figure 14-20. Basic Operation of Memory Controllers in the LBC**

Each memory bank (chip select) can be assigned to any one of these three type of machines through the machine select bits of the base register for that bank ( $BR_n[MSEL]$ ), as illustrated in [Figure 14-20](#). If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

## 14.4.1 Basic Architecture

The following sections describe the basic architecture of the LBC.

### 14.4.1.1 Address and Address Space Checking

The defined base addresses are written to the  $BR_n$  registers, while the corresponding address masks are written to the  $OR_n$  registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 19 msbs of the address, masked by  $OR_n[XAM]$  and  $OR_n[AM]$ , with the base address for each bank ( $BR_n[XBA]$  and  $BR_n[BA]$ ). If a match is found on a memory controller bank, the attributes defined in the  $BR_n$  and  $OR_n$  for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

### 14.4.1.2 External Address Latch Enable Signal (LALE)

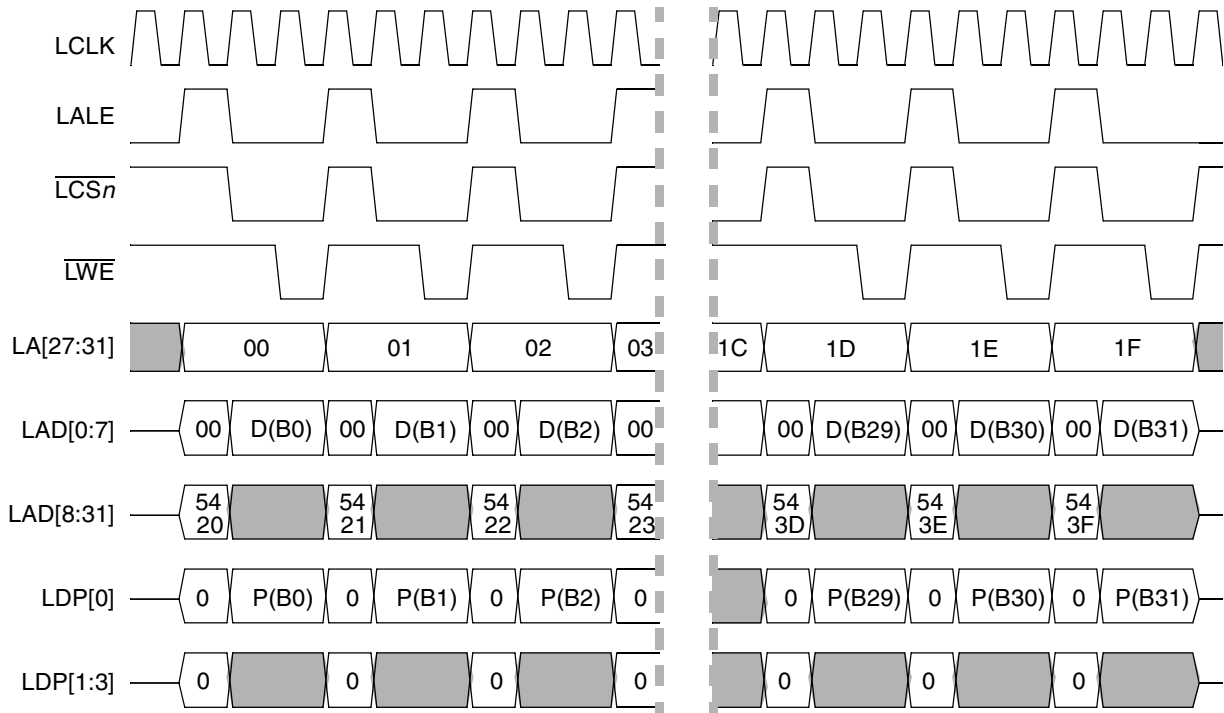
The local bus uses a multiplexed address/data bus. Therefore, the LBC must distinguish between address and data phases, which take place on the same bus (LAD[0:31] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address on the LAD[0:31] signals. An external address latch uses this signal to capture the address and provide it to the address signals of the memory or peripheral device. When LALE is negated, LAD[0:31] then serves as the (bidirectional) data bus for the access. Any address phase initiates the assertion of LALE, which has a programmable duration of between 1 and 4 bus clock cycles.

To ensure adequate hold time on the external address latch, LALE negates earlier than the address changes on LAD[0:31] during address phases. By default, LALE negates earlier by two platform clock periods (which, divided by  $LCRR[CLKDIV]$ , yields the local bus clock). For example, if the LBC is operating at 666 MHz internally, then an additional 3 ns of address hold time is introduced. However, when  $LCRR[CLKDIV] = 2$  (clock ratio of 4) and the LCLK frequency exceeds 100 MHz, the duration of the shortened LALE pulse may not meet the minimum latch enable pulse width specifications of some latches. In such cases, setting  $LBCR[AHD] = 1$  increases the LALE pulse width by one platform clock cycle, and decreases the address hold time by the same amount. At 666 MHz and with  $LCRR[CLKDIV] = 2$  (clock ratio of 4), the duration of LALE would then be 4.5 ns, with 1.5 ns of hold time. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the  $OR_n[EAD]$  and  $LCRR[EADC]$  fields, and the  $LBCR[AHD]$  bit can be left at 0. However, this will add latency to all address tenures.

The frequency of LALE assertion varies across the three memory controllers. For GPCM, every assertion of  $\overline{LCS}_n$  is considered an independent access, and accordingly, LALE asserts prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and  $\overline{LCS}_n$  32 times in order to satisfy a 32-byte cache line transfer. The SDRAM controller asserts LALE only to initiate a burst transfer with a starting address, therefore no more than one assertion of LALE may be required for SDRAM to transfer a 32-byte cache line through a 32-bit port. In the case of UPM, the frequency of LALE assertion depends on how the UPM RAM is programmed. UPM single accesses typically assert LALE once, on commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA[27:31] with and without LALE being involved. In general, when using the GPCM and SDRAM controllers it is not necessary to use LA[27:31] if a sufficiently wide latch is used to capture the entire

address during LALE phases. UPM may require LA[27:31] if the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 14-21 shows LBC signals for GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP[0] are driven with valid data and parity, respectively.



**Note:** All address and signal values are shown in hexadecimal.  
 D(Bk) = k<sup>th</sup> of 32 data bytes, P(Bk) = parity bit of k<sup>th</sup> data byte.

**Figure 14-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420**

### 14.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which the LBC asserts TA internally. In LBC debug mode, TA is also visible externally on the MDVAL signal. GPCM and SDRAM controllers automatically generate TA according to the timing parameters programmed for them in option and mode registers; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set.



Figure 14-22 shows LALE, TA (internal), and  $\overline{\text{LCS}}_n$ . Note that TA and LALE are never asserted together, and that for the duration of LALE,  $\overline{\text{LCS}}_n$  (or any other control signal) remains negated or frozen.

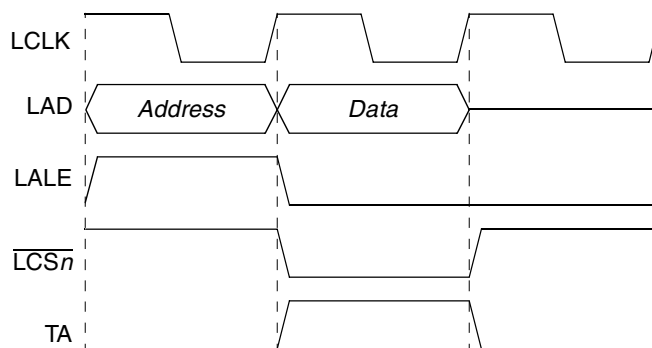


Figure 14-22. Basic LBC Bus Cycle with LALE, TA, and  $\overline{\text{LCS}}_n$

#### 14.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. LBCTL can be disabled by setting  $\text{OR}_n[\text{BCTLD}]$ . Access to an SDRAM machine controlled bank does not activate the LBCTL control. LBCTL can be further configured by  $\text{LBCR}[\text{BCTLC}]$  to act as an extra  $\overline{\text{LWE}}$  or an extra  $\overline{\text{LOE}}$  signal when in GPCM mode.

If LBCTL is configured as a data buffer control ( $\text{LBCR}[\text{BCTLC}] = 00$ ), the signal is asserted (high) on the rising edge of the bus clock on the first cycle of the memory controller operation, coincident with LALE. If the access is a write, LBCTL remains high for the whole duration. However, if the access is a read, LBCTL is negated (low) with the negation of LALE so that the memory device is able to drive the bus. If back-to-back read accesses are pending, LBCTL is asserted (high) one bus clock cycle before the next transaction starts (that is, one bus clock cycle before LALE) to allow a whole bus cycle for the bus to turn around before the next address is driven.

If an external bus transceiver is used, LBCTL should be used to signify the write direction when high. Note that the default (reset and bus idle) value of LBCTL is also high.

#### 14.4.1.5 Atomic Operation

The LBC supports the following kinds of atomic bus operations (set by  $\text{BR}_n[\text{ATOM}]$ ):

- Read-after-write atomic (RAWA). When a write access hits a memory bank in which  $\text{ATOM} = 01$ , the LBC reserves the selected memory bank for the exclusive use of the accessing master. While the bank is reserved, no other device can be granted access to this bank. The reservation is released when the master that created it accesses the same bank with a read transaction. Additional write transactions prior to the releasing read do not change reservation status, but are otherwise processed normally. If the master fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled). This feature is intended for CAM operations.
- Write-after-read atomic (WARA). When a read access hit a memory bank in which  $\text{ATOM} = 10$ , the LBC reserves the bus for the exclusive use of the accessing master.

During the reservation period, no other device can be granted access to the atomic bank. The reservation is released when the device that created it accesses the same bank with a write transaction. Additional read transactions prior to the releasing write are otherwise processed normally and do not change the reservation status. If the device fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled).

#### 14.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming  $BR_n[DECC]$ . Parity is generated and checked on a per-byte basis using  $LDP[0:3]$  for the bank if  $BR_n[DECC] = 01$  (normal parity) or  $BR_n[DECC] = 10$  for read-modify-write (RMW) parity. Byte lane parity on  $LDP[0:3]$  is generated regardless of the  $BR_n[DECC]$  setting. Note that RMW parity can be used only for 32-bit port size banks.  $LBCR[EPAR]$  determines the global type of parity (odd or even).

#### 14.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor starts counting down from the time-out value ( $LBCR[BMT]$ ) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the countdown until the data tenure completes and then idles if there is no pending transaction. Setting  $LTEDR[BMD]$  disables bus monitor error checking (i.e., the  $LTESR[BM]$  bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in [Section 14.4.4.1.4, “Exception Requests”](#)) or terminate a GPCM access.

It is very important to ensure that the value of  $LBCR[BMT]$  is not set too low; otherwise spurious bus time-outs may occur during normal operation—particularly for SDRAMs—resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles),  $LBCR[BMT]$  must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

### 14.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPRM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups— $BR_n$  and  $OR_n$ .

[Figure 14-23](#) shows a simple connection between an 8-bit port size SRAM device and the LBC in GPCM mode. Byte-write enable signals ( $\overline{LWE}$ ) are available for each byte written to memory. Also, the output enable signal ( $\overline{LOE}$ ) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ( $\overline{LCS0}$ ) prior to the system being fully configured.

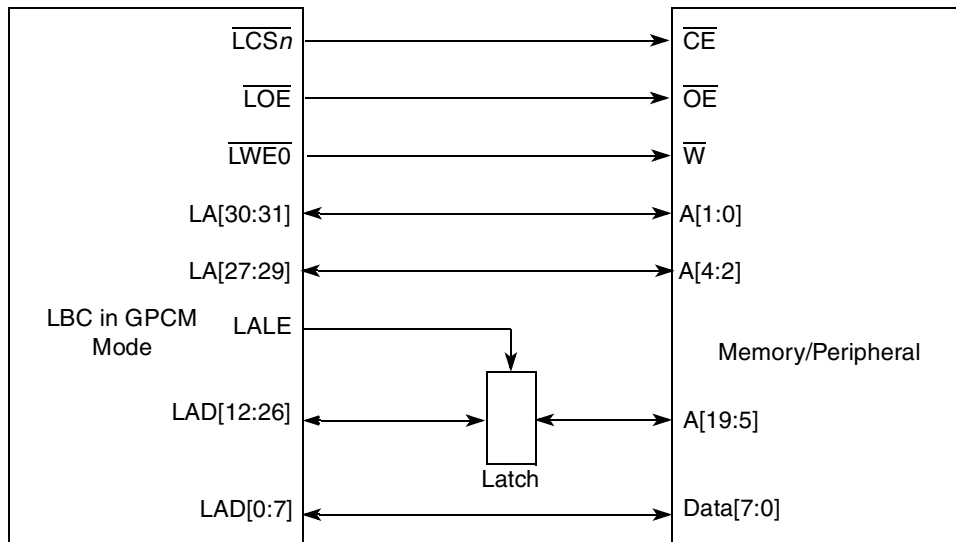


Figure 14-23. Local Bus to GPCM Device Interface

Figure 14-24 shows  $\overline{LCS}$  as defined by the setup time required between the address lines and  $\overline{LCS}$ . The user can configure  $ORn[ACS]$  to specify  $\overline{LCS}$  to meet this requirement.

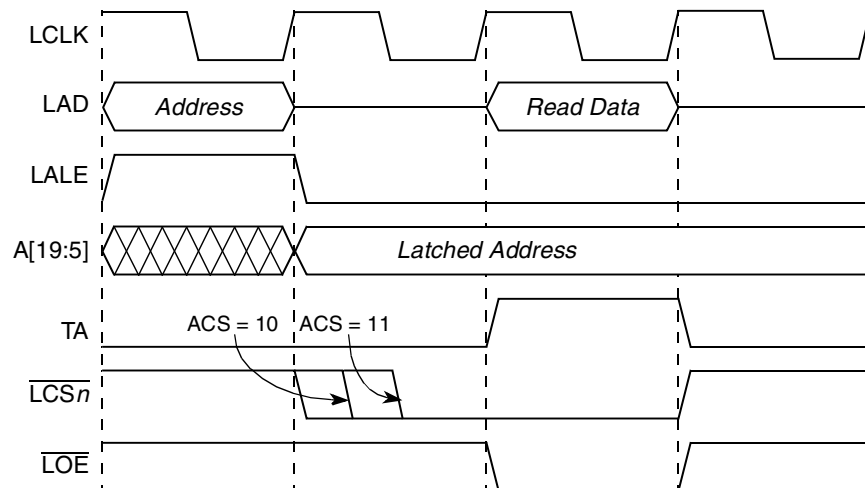


Figure 14-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0)

### 14.4.2.1 Timing Configuration

If  $BRn[MSEL]$  selects the GPCM, the attributes for the memory cycle are taken from  $ORn$ . These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR, and SETA fields.

Table 14-23 shows signal behavior and system response for a write access.

**Table 14-23. GPCM Write Control Signal Timing**

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}}_n$ Asserted	$\overline{\text{LCS}}_n$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles <sup>1</sup>
0	0	00	0	0	0	0	3+SCY
0	0	10	0	1/4	0	0	3+SCY
0	0	11	0	1/2	0	0	3+SCY
0	1	00	0	0	0	0	3+SCY
0	1	10	0	1	0	0	3+SCY
0	1	11	0	2	0	0	4+SCY
0	0	00	1	0	0	-1/4	3+SCY
0	0	10	1	1/4	-1/4	-1/4	3+SCY
0	0	11	1	1/2	-1/4	-1/4	3+SCY
0	1	00	1	0	0	-1/4	3+SCY
0	1	10	1	1	-1/4	-1/4	3+SCY
0	1	11	1	2	-1/4	-1/4	4+SCY
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/4	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	-1-1/4	4+2*SCY
1	0	10	1	1+1/4	-1-1/4	-1-1/4	5+2*SCY
1	0	11	1	1+1/2	-1-1/4	-1-1/4	5+2*SCY
1	1	00	1	0	0	-1-1/4	4+2*SCY
1	1	10	1	2	-1-1/4	-1-1/4	5+2*SCY
1	1	11	1	3	-1-1/4	-1-1/4	6+2*SCY

<sup>1</sup> Total cycles when LALE is asserted for one cycle only ( $\text{OR}_n[\text{EAD}] = 0$ ;  $\text{OR}_n[\text{EAD}] = 1$  and  $\text{LCRR}[\text{EADC}] = 01$ ). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 14-24 shows the signal behavior and system response for a read access.

**Table 14-24. GPCM Read Control Signal Timing**

Option Register Attributes				Signal Behavior (Bus Clock Cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCSn}}$ Asserted	$\overline{\text{LCSn}}$ Negated to Address Change	Total Cycles <sup>1</sup>
0	0	0	00	0	1	4+SCY
0	0	0	10	1/4	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/4	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/4	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/4	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

<sup>1</sup> Total cycles when LALE is asserted for one cycle only ( $\text{ORn[EAD]} = 0$ ;  $\text{ORn[EAD]} = 1$  and  $\text{LCRR[EADC]} = 01$ ). Asserting LALE for more than one cycle increases the total cycle count accordingly.

### 14.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM support an option to drive the  $\overline{\text{LCSn}}$  signal with different timings (with respect to the external address/data bus).  $\overline{\text{LCSn}}$  can be driven in any of the following ways:

- Simultaneous with the latched memory address. (This refers to the externally latched address, not the address timing on LAD[0:31]. That is, chip select does not assert during LALE).

- One quarter of a clock cycle later.
- One half of a clock cycle later
- One clock cycle later (for  $LCRR[CLKDIV] = 2$  (clock ratio of 4)) when  $ORn[XACS] = 1$ .
- Two clock cycles later, when  $ORn[XACS] = 1$ .
- Three clock cycles later, when  $ORn[XACS] = 1$  and  $ORn[TRLX] = 1$ .

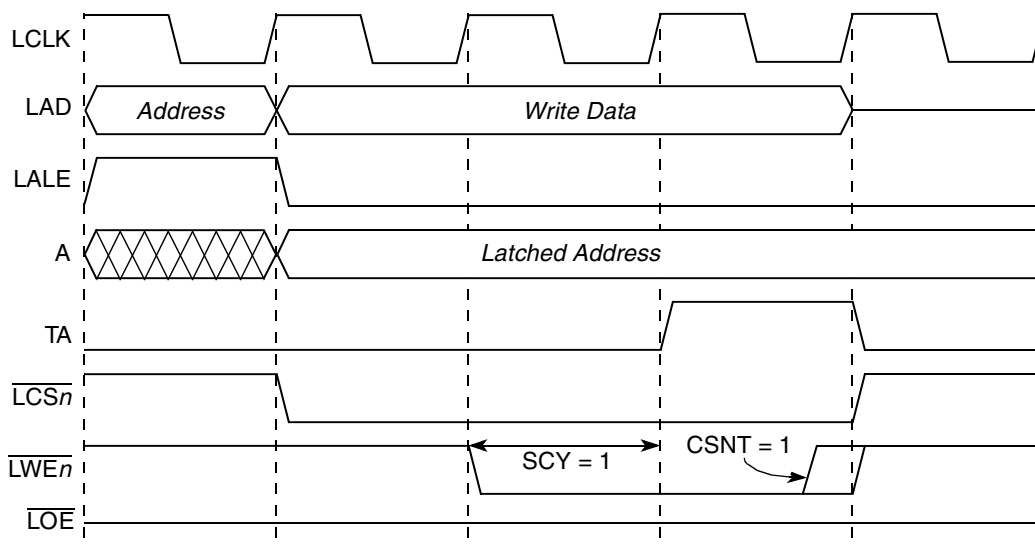
The timing diagram in [Figure 14-24](#) shows two chip-select assertion timings.

#### 14.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between zero and 30 wait states to be added to an access by programming  $ORn[SCY]$  and  $ORn[TRLX]$ . Internal generation of transfer acknowledge is enabled if  $ORn[SETA] = 0$ . If  $\overline{LGTA}$  is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by  $\overline{LGTA}$ ; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of  $ORn[SETA]$ , wait states prolong the assertion duration of both  $\overline{LOE}$  and  $\overline{LWE}n$  in the same manner. When  $TRLX = 1$ , the number of wait states inserted by the memory controller is doubled from  $ORn[SCY]$  cycles to  $2 \times ORn[SCY]$  cycles, allowing a maximum of 30 wait states.

#### 14.4.2.2.2 Chip-Select and Write Enable Negation Timing

[Figure 14-23](#) shows a basic connection between the local bus and a static memory device. In this case,  $\overline{LCS}n$  is connected directly to  $\overline{CE}$  of the memory device. The  $\overline{LWE}[0:3]$  signals are connected to the respective  $\overline{WE}[3:0]$  signals on the memory device where each  $\overline{LWE}[0:3]$  signal corresponds to a different data byte.



**Figure 14-25. GPCM Basic Write Timing ( $XACS = 0$ ,  $ACS = 00$ ,  $CSNT = 1$ ,  $SCY = 1$ ,  $TRLX = 0$ )**

As [Figure 14-25](#) shows, the timing for  $\overline{LCS}n$  is the same as for the latched address. The strobes for the transaction are supplied by  $\overline{LOE}$  or  $\overline{LWE}n$ , depending on the transaction direction—read or write (write case shown in [Figure 14-25](#)).  $ORn[CSNT]$  controls the timing for the appropriate strobe negation in write cycles.

When this attribute is asserted, the strobe is negated one quarter of a clock before the normal case. For example, when  $ACS = 00$  and  $CSNT = 1$ ,  $\overline{LWE}n$  is negated one quarter of a clock earlier, as shown in Figure 14-25.

### 14.4.2.2.3 Relaxed Timing

$ORx[TRLX]$  is provided for memory systems that require more relaxed timing between signals. Setting  $TRLX = 1$  has the following effect on timing:

- An additional bus cycle is added between the address and control signals (but only if  $ACS \neq 00$ ).
- The number of wait states specified by  $SCY$  is doubled, providing up to 30 wait states.
- The extended hold time on read accesses ( $EHTR$ ) is extended further.
- $\overline{LCS}n$  signals are negated 1 cycle earlier during writes (but only if  $ACS \neq 00$ ).
- $\overline{LWE}[0:3]$  signals are negated 1 cycle earlier during writes.

Figure 14-26 and Figure 14-27 show relaxed timing read and write transactions. The example in Figure 14-27 also shows address and data multiplexing on  $LAD[0:31]$  for a pair of writes issued consecutively.

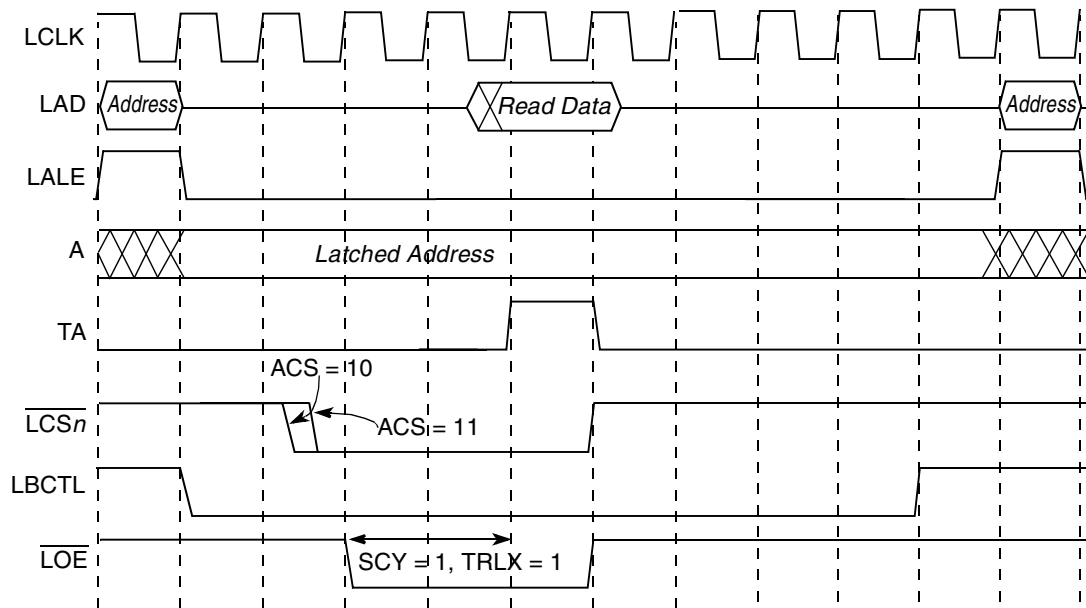
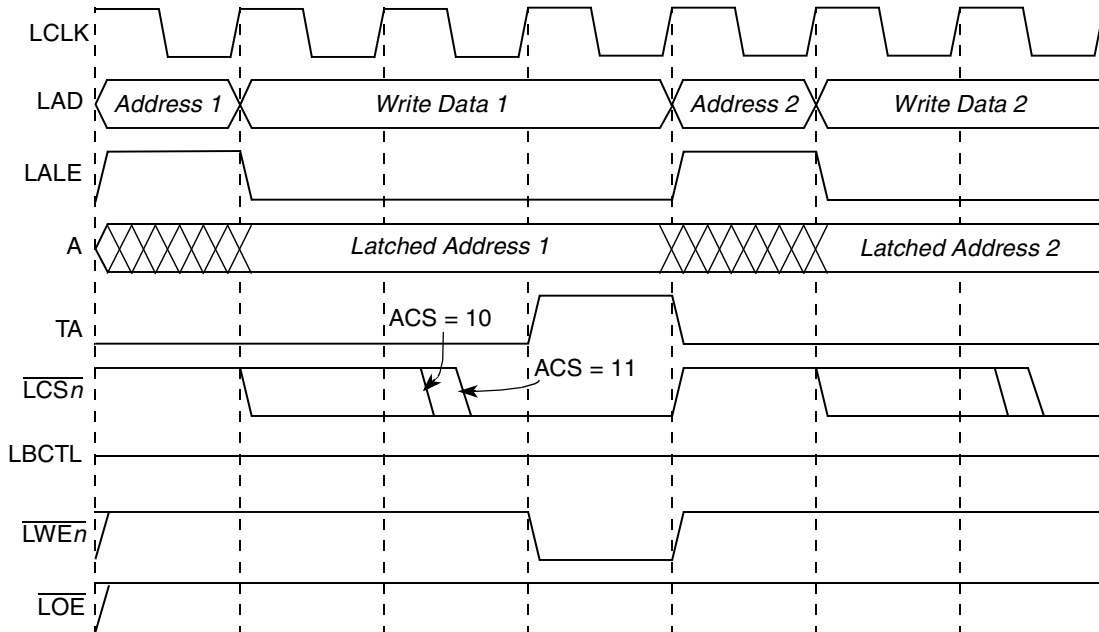
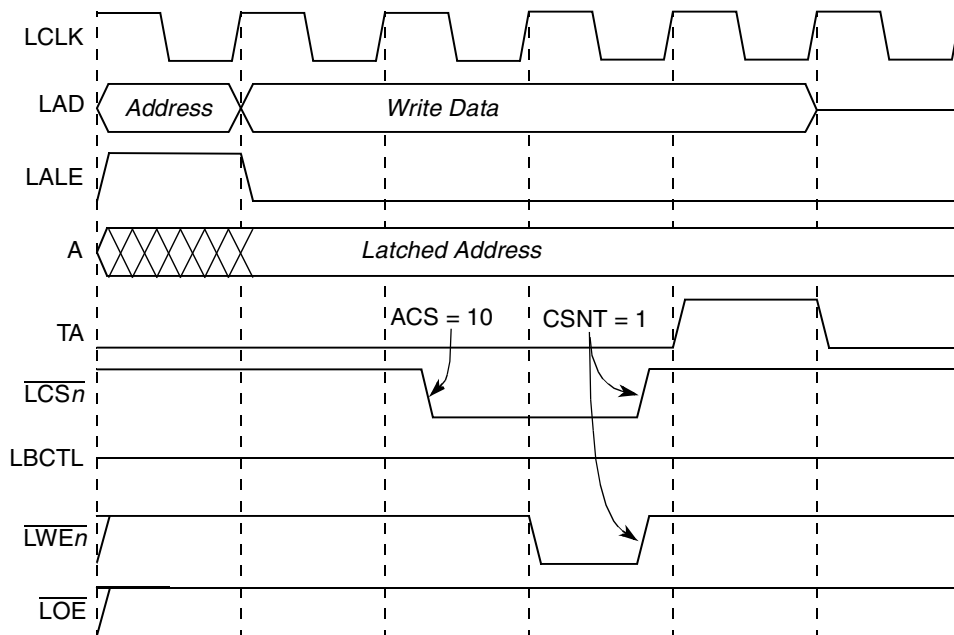


Figure 14-26. GPCM Relaxed Timing Read ( $XACS = 0$ ,  $ACS = 1x$ ,  $SCY = 1$ ,  $EHTR = 0$ ,  $TRLX = 1$ )



**Figure 14-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1)**

When TRLX and CSNT are set in a write access, the  $\overline{LWE}[0:3]$  strobe signals are negated one clock earlier than in the normal case, as shown in Figure 14-28 and Figure 14-29. If  $ACS \neq 00$ ,  $\overline{LCSn}$  is also negated one clock earlier.



**Figure 14-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)**



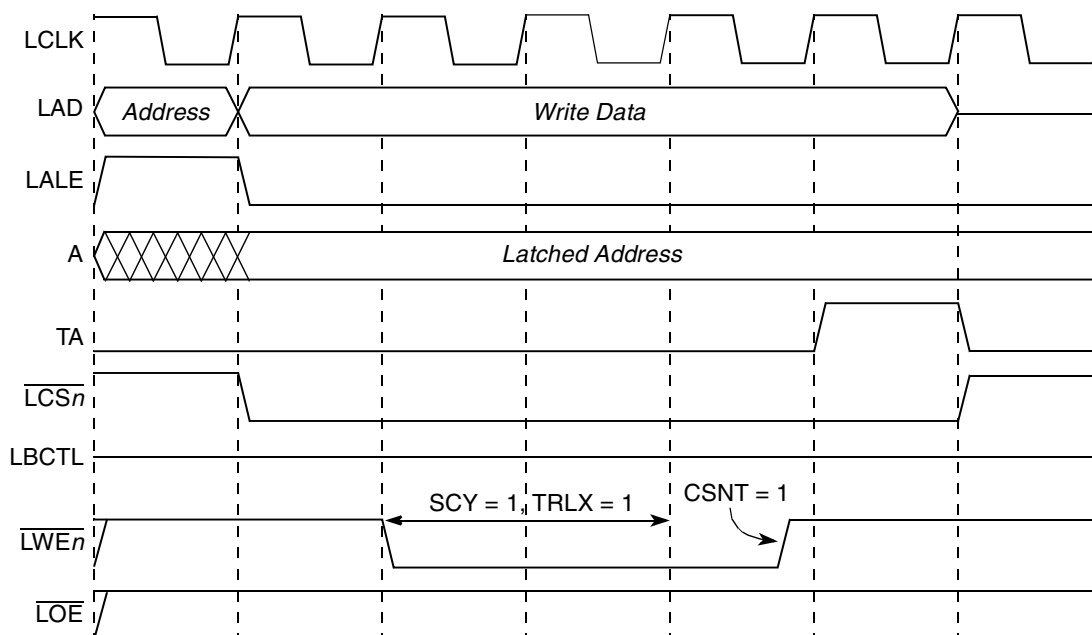


Figure 14-29. GPCM Relaxed Timing Write  
(XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1)

#### 14.4.2.2.4 Output Enable ( $\overline{LOE}$ ) Timing

The timing of  $\overline{LOE}$  is affected only by TRLX. It always asserts and negates on the rising edge of the bus clock.  $\overline{LOE}$  asserts either on the rising edge of the bus clock after  $\overline{LCSn}$  is asserted or coinciding with  $\overline{LCSn}$  (if XACS = 1 and ACS = 10 or 11). Accordingly, assertion of  $\overline{LOE}$  can be delayed (along with the assertion of  $\overline{LCSn}$ ) by programming TRLX = 1.  $\overline{LOE}$  negates on the rising clock edge coinciding with  $\overline{LCSn}$  negation

#### 14.4.2.2.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of  $ORn[TRLX,EHTR]$ . Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified by the configuration of  $ORn[TRLX,EHTR]$ , as described in Section 14.3.1.2.2, “Option Registers ( $ORn$ )—GPCM Mode,” in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the LBC for reads, regardless of the setting of  $ORn[EHTR]$ . Figure 14-30, Figure 14-31, and Figure 14-32 present various GPCM timing examples.

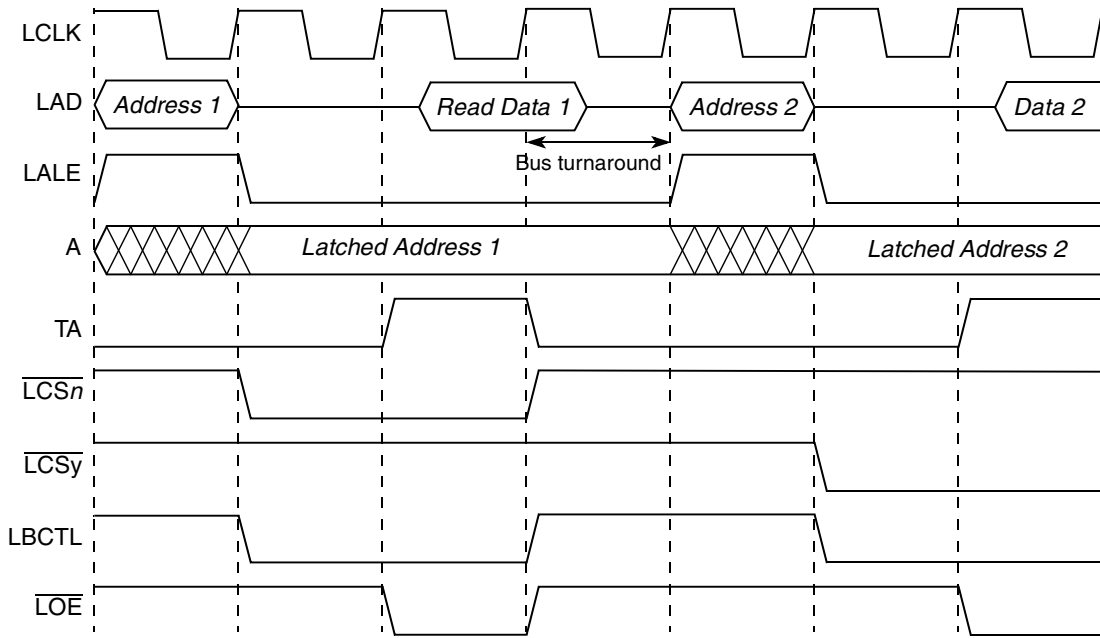


Figure 14-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

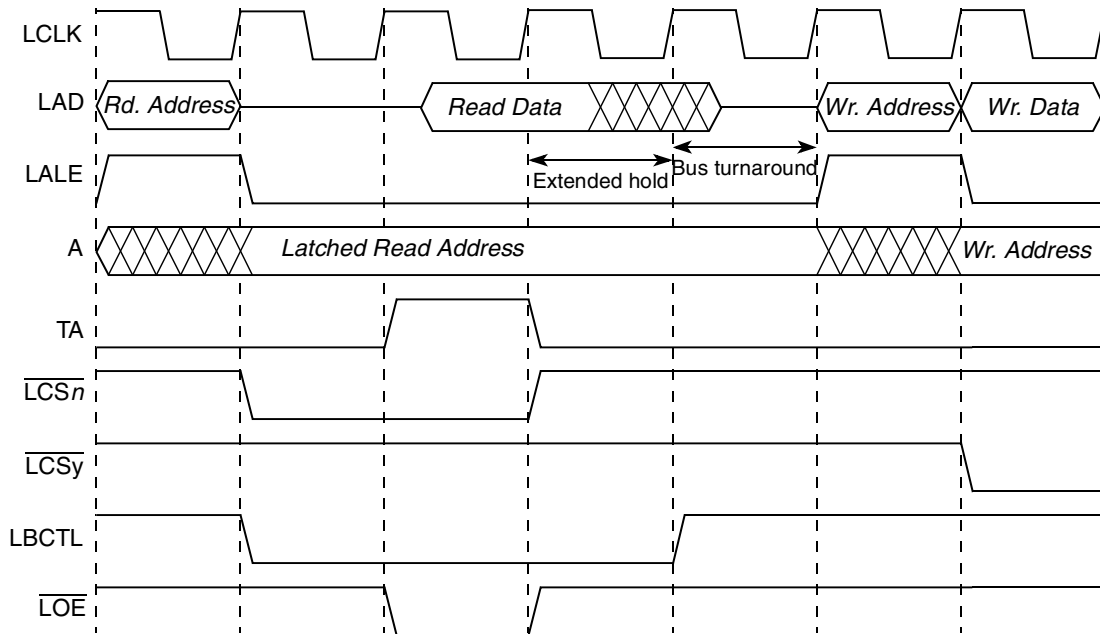
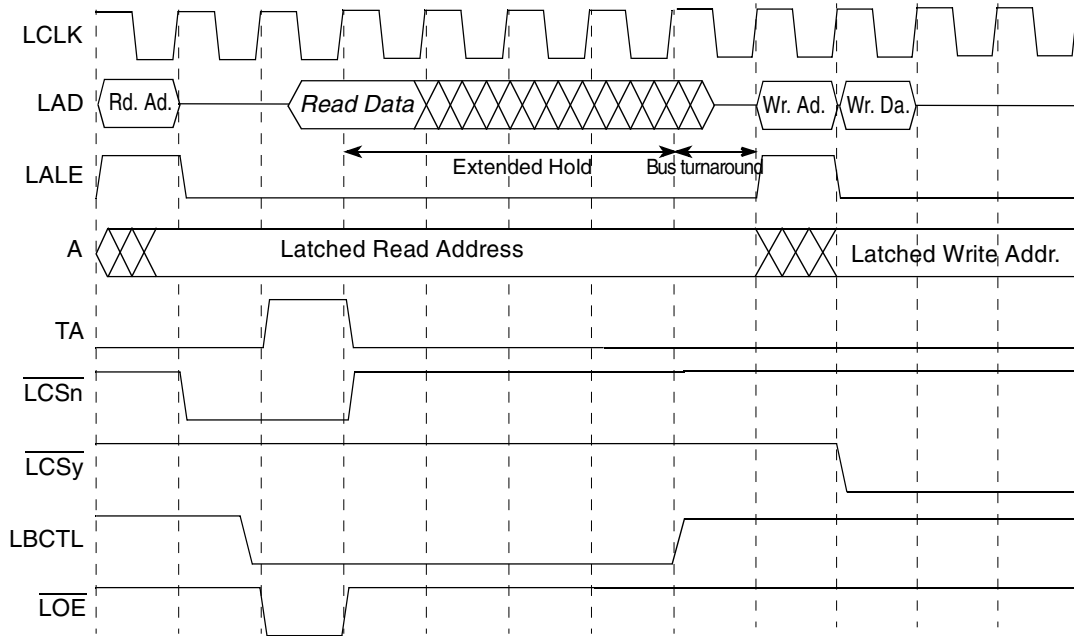


Figure 14-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads)



**Figure 14-32. GPCM Read Followed by Write**  
**(TRLX = 1, EHTR = 0, 4-Cycle Extended Hold Time on Reads)**

### 14.4.2.3 External Access Termination ( $\overline{\text{LGTA}}$ )

External access termination is supported by the GPCM using the asynchronous  $\overline{\text{LGTA}}$  input signal, which is synchronized and sampled internally by the local bus. If, during assertion of  $\overline{\text{LCS}}_n$ , the sampled  $\overline{\text{LGTA}}$  signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of  $\text{OR}_n[\text{SETA}]$ ).  $\overline{\text{LGTA}}$  should be asserted for at least one bus cycle to be effective. Note that because  $\overline{\text{LGTA}}$  is synchronized, bus termination occurs two cycles after  $\overline{\text{LGTA}}$  assertion, so in the case of a read cycle, the device still must drive data as long as  $\overline{\text{LOE}}$  is asserted.

The user selects whether transfer acknowledge is generated internally or externally ( $\overline{\text{LGTA}}$ ) by programming  $\text{OR}_n[\text{SETA}]$ . Asserting  $\overline{\text{LGTA}}$  always terminates an access, even if  $\text{OR}_n[\text{SETA}] = 0$  (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if  $\text{OR}_n[\text{SETA}] = 1$ . The timing of  $\overline{\text{LGTA}}$  is illustrated by the example in [Figure 14-33](#).

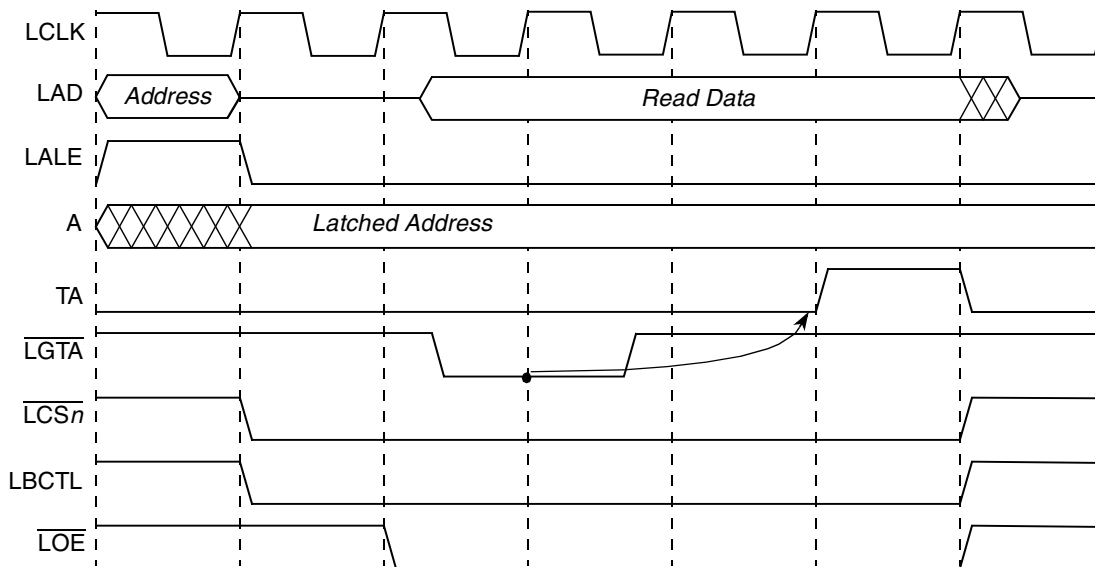


Figure 14-33. External Termination of GPCM Access

### 14.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization.  $\overline{\text{LCS}}_0$  is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset,  $\overline{\text{LCS}}_0$  is asserted for every local bus access until  $\text{BR}_0$  or  $\text{OR}_0$  is reconfigured.

The boot chip-select also provides a programmable port size, which is configured during reset. The boot chip-select does not provide write protection.  $\overline{\text{LCS}}_0$  operates this way until the first write to  $\text{OR}_0$  and it can be used as any other chip-select register after the preferred address range is loaded into  $\text{BR}_0$ . After the first write to  $\text{OR}_0$ , the boot chip-select can be restarted only with a hardware reset. [Table 14-25](#) describes the initial values of the boot bank in the memory controller.

Table 14-25. Boot Bank Field Values After Reset

Register	Field	Setting	Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0	OR0	AM	0000_0000_0000_0000_0
	XBA	00		XAM	00
	PS	From signal during reset.		BCTLD	0
	DECC	00		CSNT	1
	WP	0		ACS	11
	MSEL	000		XACS	1
	ATOM	00		SCY	1111
	V	1		SETA	0
			TRLX	1	
			EHTR	1	
			EAD	1	

### 14.4.3 SDRAM Machine

The LBC provides an SDRAM interface (machine) for the local bus. The machine provides the control functions and signals for Intel PC133 and JEDEC-compliant SDRAM devices. Each bank can control an SDRAM device on the local bus.

#### 14.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR).

Figure 14-34 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[2:0] of the SDRAM connect directly to LA[27:29], address signal A10 connects to the LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

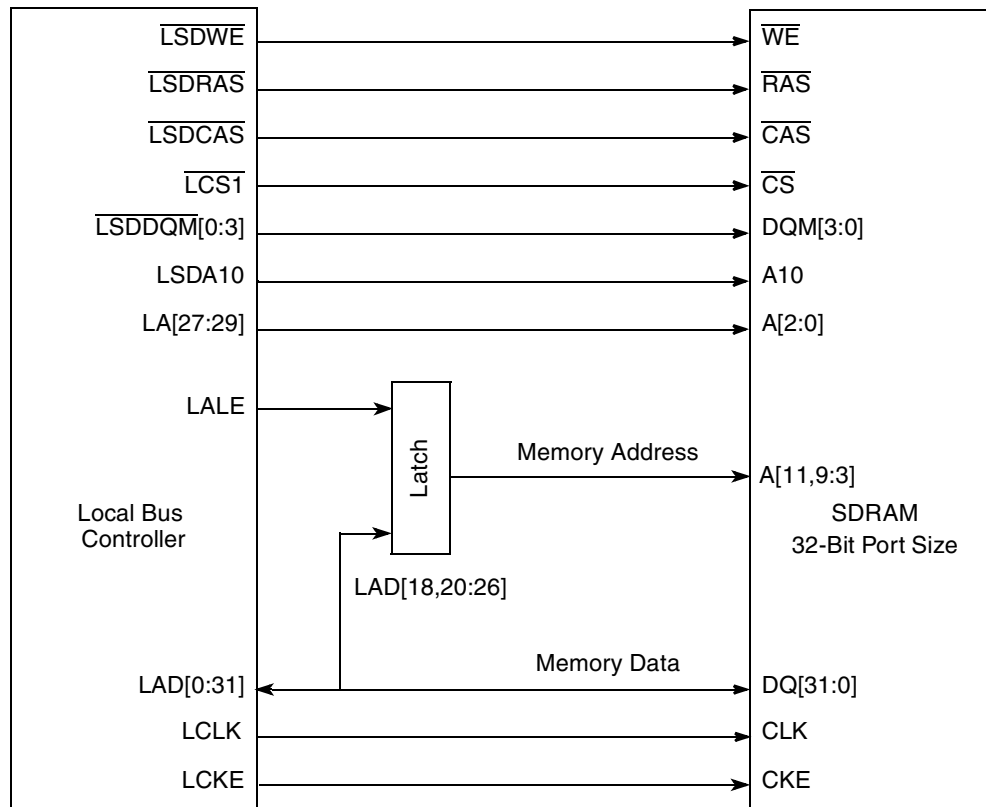


Figure 14-34. Connection to a 32-Bit SDRAM with 12 Address Lines

### 14.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers ( $\text{OR}_n$ ,  $\text{BR}_n$ ,  $\text{LSDMR}$ ). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

- Issue a PRECHARGE-ALL-BANKS command
- Issue eight AUTO-REFRESH commands
- Issue a MODE-SET command to initialize the mode register

The initial commands are executed by setting  $\text{LSDMR}[\text{OP}]$  and accessing the SDRAM with any write that hits the relevant bank. Since the result of any update to the  $\text{LSDMR}$  must be in effect before accessing the SDRAM with any write, a write to  $\text{LSDMR}$  should be followed immediately by a read from  $\text{LSDMR}$ , which must complete prior to an initial write to SDRAM. Further, the first write to SDRAM should be followed immediately by an SDRAM read, which must complete prior to additional  $\text{LSDMR}$  updates. This enforces a proper ordering between updates to the  $\text{LSDMR}$  and write accesses to the SDRAM. If the initialization is being done by the e500, this described protocol is guaranteed only if the SDRAM is mapped as cache-inhibited and guarded, as the CCSR memory region containing  $\text{LSDMR}$  should be. If the initialization is from an external host, said host must ensure completion of  $\text{LSDMR}$  and SDRAM reads prior to subsequent writes, as described above.

Note that software should ensure that no memory operations begin until this process completes.

**NOTE**

In general (not only during power-on reset) the LSDMR/SDRAM access ordering protocol should be observed for proper operation.

**14.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands**

The SDRAM machine performs all accesses to SDRAM by using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

The following SDRAM interface commands are provided by setting LSDMR[OP] to a non-zero value (LSDMR[OP] = 000 sets normal read/write operation):

**Table 14-26. SDRAM Interface Commands**

<b>Command (LSDMR[OP])</b>	<b>Description</b>
ACTIVATE (110)	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another ACTIVATE is issued.
MODE-SET (011)	Allows setting of SDRAM options—CAS latency and burst length. CAS latency depends on the SDRAM device used. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the local bus memory controller supports only 8-beat bursts for 8-bit and 32-bit port size, or 4-beat bursts for 16-bit port size. The LBC does not support burst lengths of 1, 2 and a page for SDRAMs. The mode register data (CAS latency and burst length) is programmed into the LSDMR register by initialization software after reset. After the LSDMR is set, the LBC transfers the information to the SDRAM device by issuing a MODE-SET command.
PRECHARGE (100: single bank) (101: all-banks)	Restores data from the sense amplifiers to the appropriate row in the SDRAM device array. Also initializes the sense amplifiers to prepare for activating another row in the SDRAM device. Note that the LBC uses LSDA10 to distinguish between PRECHARGE-ALL-BANKS (LSDA10 is high) and PRECHARGE-SINGLE-BANK (LSDA10 is low). The SDRAMs must be compatible with this format.
READ (111)	Latches the column address and transfers data from the selected sense amplifier on the SDRAM device, to the output buffer as determined by the column address. During each successive clock, additional data is driven without additional read commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. Read data is discarded by the LBC.
WRITE (111)	Latches the column address and transfers data from the data signals to the selected sense amplifier on the SDRAM device, as determined by the column address. During each successive clock, additional data is transferred to the sense amplifiers from the data signals without additional write commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. LSDDQM[0:3] are inactive and write data is undefined.
AUTO-REFRESH (001)	Causes a row to be read in all memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After being read, a row is automatically rewritten into the memory array. All banks must be in a precharged state before executing refresh.
SELF-REFRESH (010)	Allows data to be retained in the SDRAM device, even when the rest of the LBC is in a power saving mode with clocks turned off. When placed in this mode, the SDRAM device is capable of issuing its own refresh commands, without external clocking from the LBC and the LCKE signal from the LBC is negated. This command can be issued at any time. Normal operation can be resumed only by setting LSDMR[OP] = 000, and waiting a minimum of 200 bus cycles before issuing reads or writes to the LBC.

#### 14.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the  $OR_n$  register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found, together with a bank match, the bus cycle is defined as a page hit. An open page is automatically closed by the SDRAM machine if the bus becomes idle, unless  $OR_n[PMSEL] = 1$ .

#### 14.4.3.5 Page Management

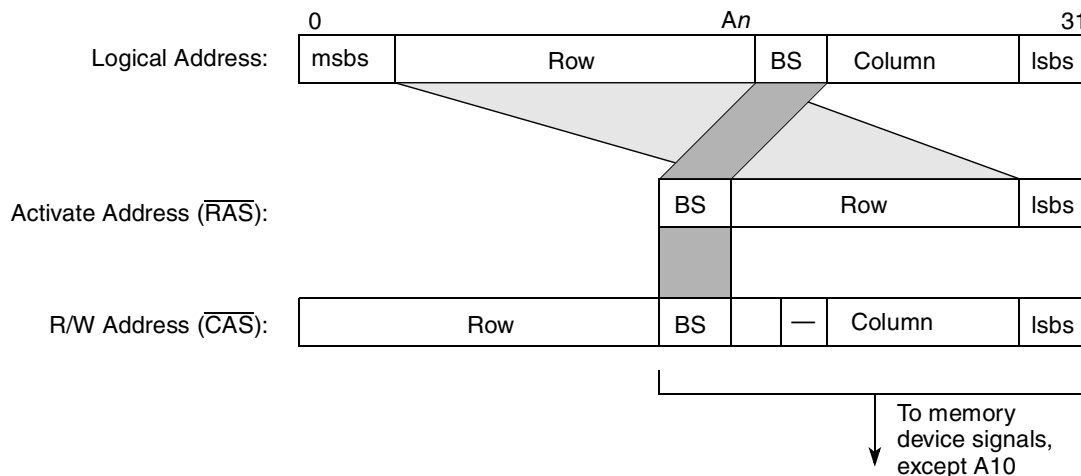
The LBC can manage at most four open pages (one page per SDRAM bank) for a single SDRAM device. After a page is opened, it remains open unless:

- The next access is to a page in a different SDRAM device, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The next access is to a page in an SDRAM bank that has a different page open on it, in which case the old page is closed with a PRECHARGE-SINGLE-BANK command.
- The current SDRAM device requires refresh services, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The bus becomes idle and  $OR_n[PMSEL] = 0$ , in which case all open pages in the current device are closed with a PRECHARGE-ALL-BANKS command.

#### 14.4.3.6 SDRAM Address Multiplexing

The lower address bus bits are connected to the memory device's address port with the memory controller multiplexing the row/column and the internal bank select lines. The position of the bank select lines are set according to  $LSDMR[BSMA]$ . [Figure 14-35](#) shows how the SDRAM controller shifts the row address down to the lower output address signals during activate and shifts the bank select bits up to the address signals specified by  $LSDMR[BSMA]$ , supporting page-based interleaving. The lsb of the logical row address ( $A_n$  in [Figure 14-35](#)) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).





**Figure 14-35. SDRAM Address Multiplexing**

Note that during normal operation (read/write), a full 32-bit address that includes row and column is generated on LAD[0:31]. However, address/data signal multiplexing implies that the address must be latched by an external latch that is controlled by LALE. All SDRAM device address signals need to be connected to the latched address bits and burst address bits (LA[27:31]) of the LBC, with the exception of A10, which has a dedicated connection on LSDA10. LSDA10 is driven with the appropriate row address bit for SDRAM commands that require A10 to be an address.

#### 14.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the OR $_n$  and LSDMR registers. These parameters include the following:

- Precharge to activate interval (LSDMR[PRETOACT])
- Activate to read/write interval (LSDMR[ACTTORW])
- CAS latency, column address to first data out (LSDMR[CL] and LCRR[ECL])
- Write recovery, last data in to precharge (LSDMR[WRC])
- Refresh recovery interval (LSDMR[RFRC])
- External buffers on the control lines present (LSDMR[BUFCMD] and LCRR[BUFCMDC])

In addition, the LBC hardware ensures a default activate to precharge interval of 10 bus cycles. The following sections describe SDRAM parameters programmed in LSDMR.

### 14.4.3.7.1 Precharge-to-Activate Interval

The precharge-to-activate interval parameter, controlled by LSDMR[PRETOACT], defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

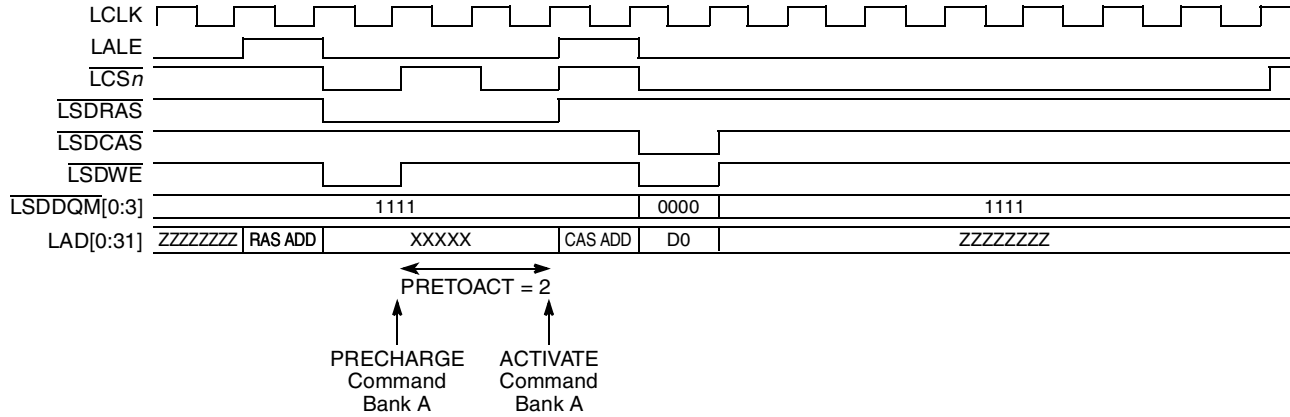


Figure 14-36. PRETOACT = 2 (2 Clock Cycles)

### 14.4.3.7.2 Activate-to-Read/Write Interval

This parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

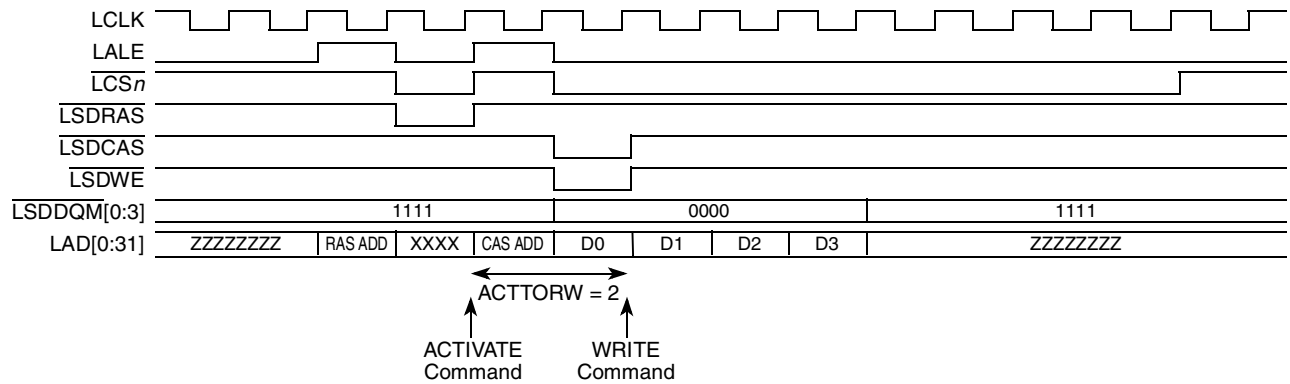


Figure 14-37. ACTTORW = 2 (2 Clock Cycles)

### 14.4.3.7.3 Column Address to First Data Out—CAS Latency

This parameter, controlled by LSDMR[CL] for latency of 1, 2, or 3 and by LCRR[ECL] for latency of more than 3, defines the timing for first read data after a column address is sampled by the SDRAM.

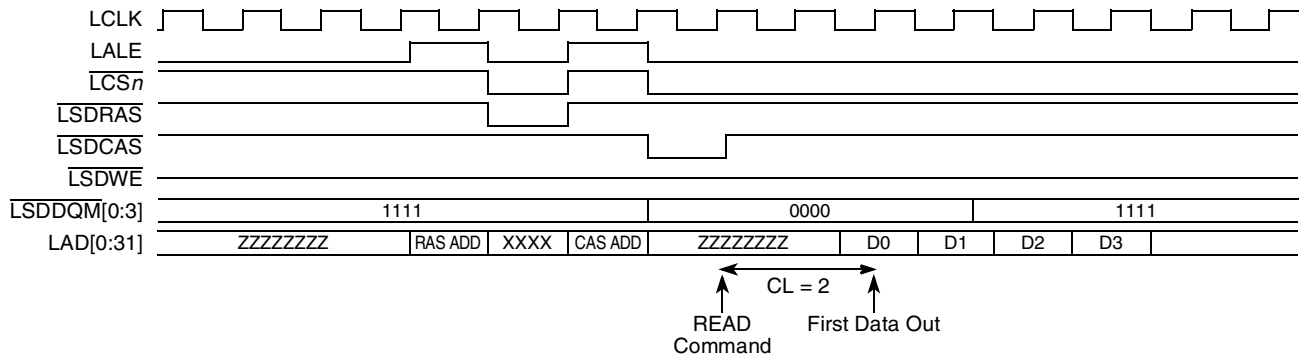


Figure 14-38. CL = 2 (2 Clock Cycles)

### 14.4.3.7.4 Last Data In to Precharge—Write Recovery

This parameter, controlled by LSDMR[WRC], defines the earliest timing for a PRECHARGE command after the last data was written to the SDRAM.

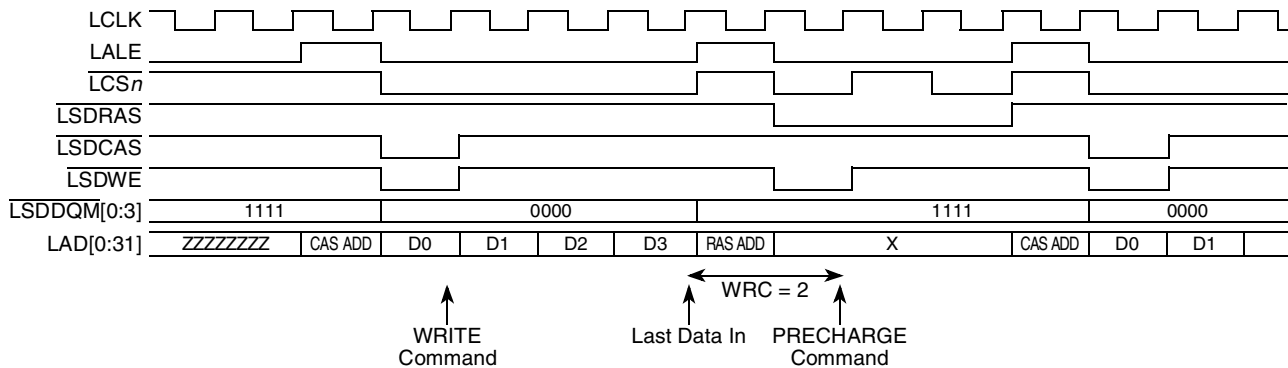


Figure 14-39. WRC = 2 (2 Clock Cycles)

### 14.4.3.7.5 Refresh Recovery Interval (RFRC)

This parameter, controlled by LSDMR[RFRC], defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command to the same SDRAM device.

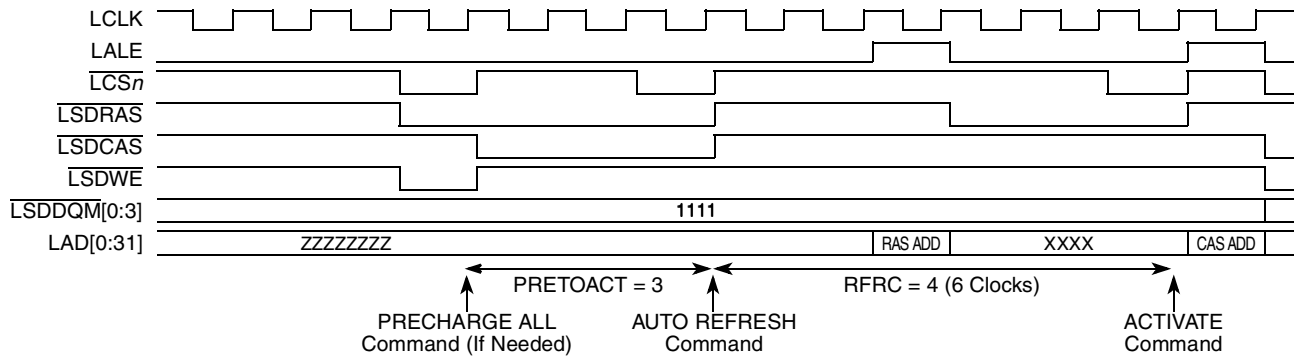


Figure 14-40. RFRC = 4 (6 Clock Cycles)

### 14.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ( $\overline{\text{LSDRAS}}$ ,  $\overline{\text{LSDCAS}}$ ,  $\overline{\text{LSDWE}}$ , and  $\text{LSDA10}$ ), is endangering the device setup time, LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add LCCR[BUFCMDC] extra bus cycles to the assertion of SDRAM control signals ( $\overline{\text{LSDRAS}}$ ,  $\overline{\text{LSDCAS}}$ ,  $\overline{\text{LSDWE}}$ , and  $\text{LSDA10}$ ) for each SDRAM command.

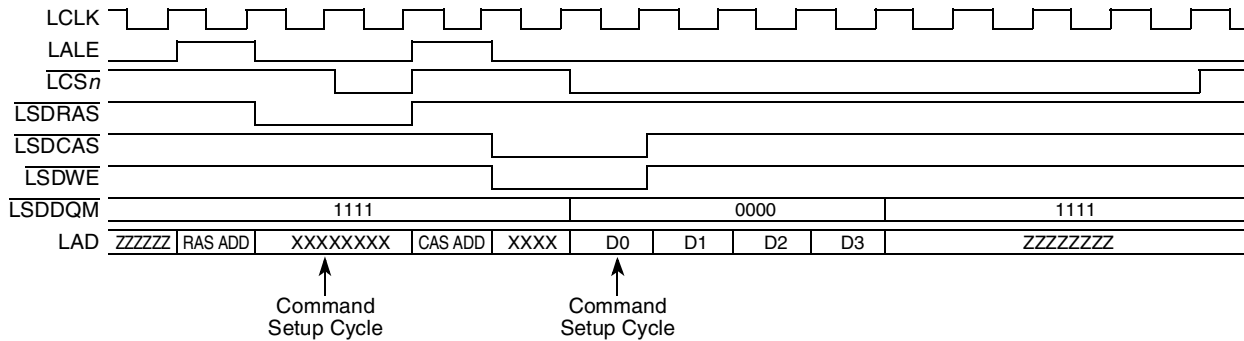


Figure 14-41. BUFCMD = 1, LCCR[BUFCMDC] = 2

### 14.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

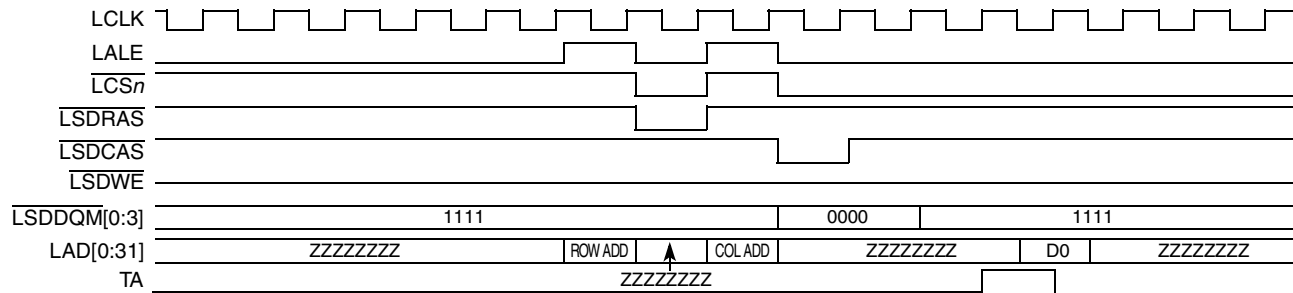


Figure 14-42. SDRAM Single-Beat Read, Page Closed, CL = 3

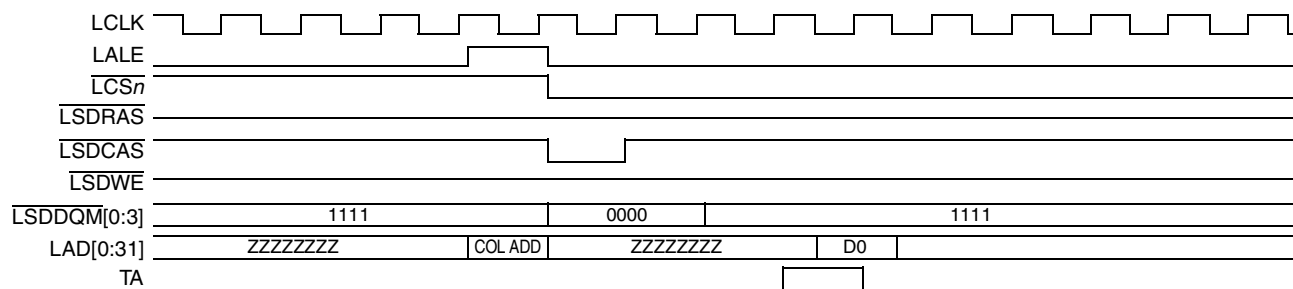


Figure 14-43. SDRAM Single-Beat Read, Page Hit, CL = 3

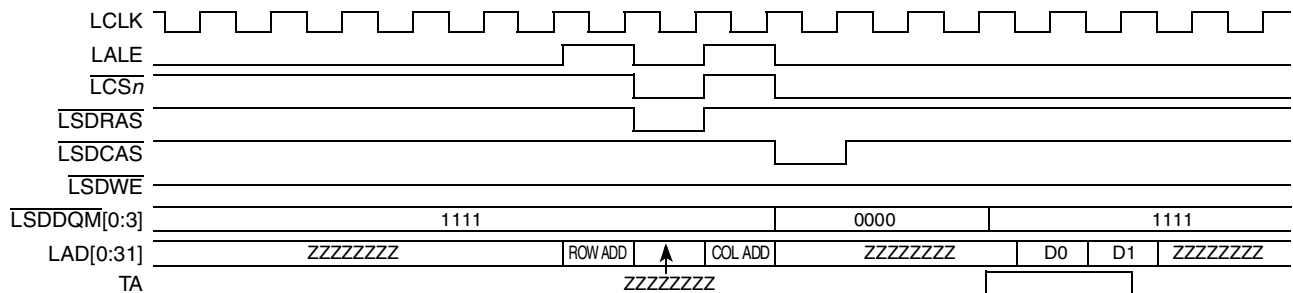


Figure 14-44. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

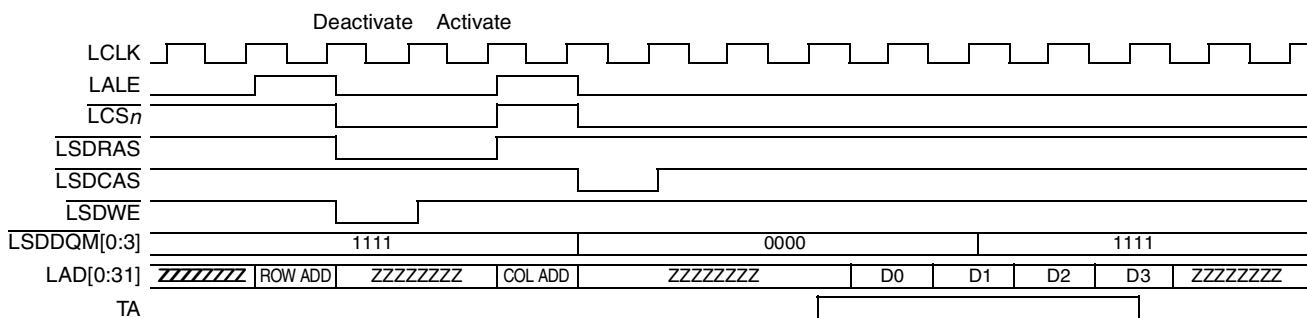


Figure 14-45. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

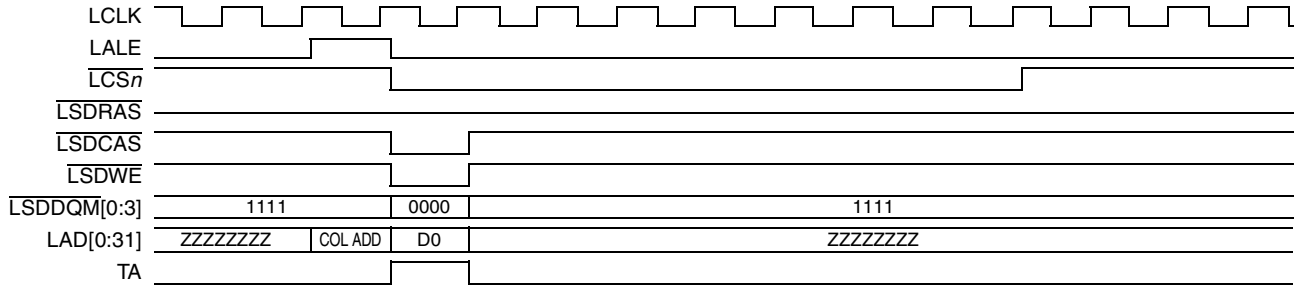


Figure 14-46. SDRAM Single-Beat Write, Page Hit

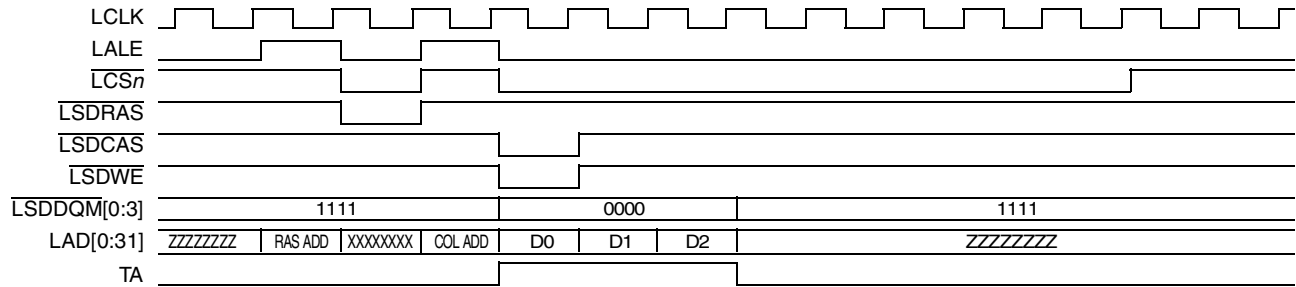


Figure 14-47. SDRAM Three-Beat Write, Page Closed

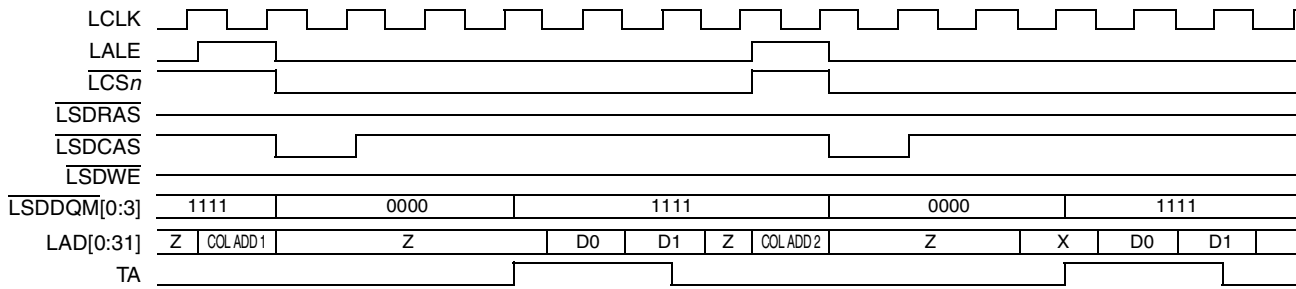


Figure 14-48. SDRAM Read-After-Read Pipelined, Page Hit, CL = 3

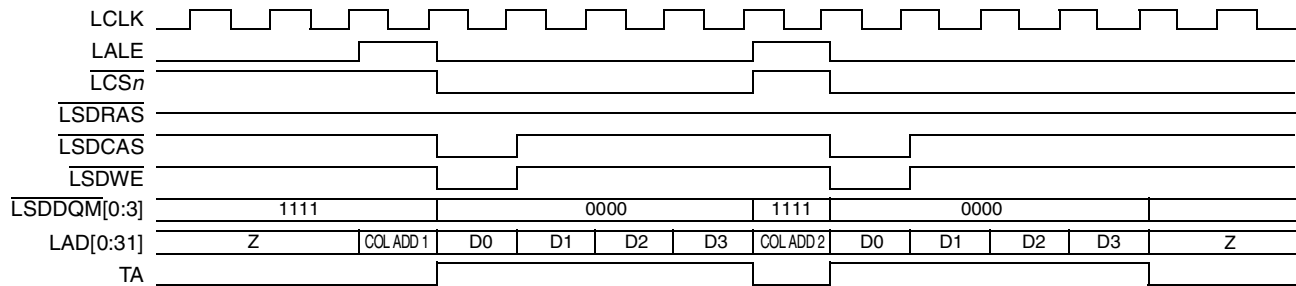


Figure 14-49. SDRAM Write-After-Write Pipelined, Page Hit

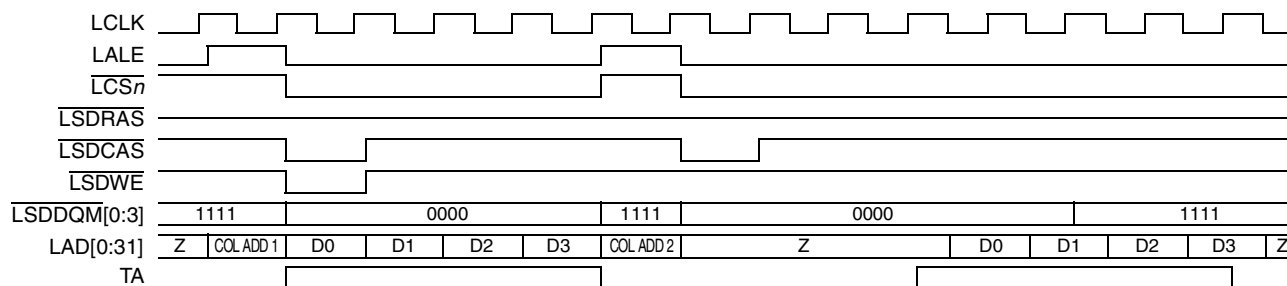


Figure 14-50. SDRAM Read-After-Write Pipelined, Page Hit

### 14.4.3.9 SDRAM Read/Write Transactions

The SDRAM interface supports read and write transactions of between 1 and 8 data beats for transaction sizes ranging from 1 to 32 bytes. A full burst is performed for each transaction, with the burst length dependent on the port size. A maximum burst of 8 beats is used for an 8-bit or 32-bit port size, while a maximum burst of 4 beats is used for a 16-bit port size, as programmed in LSDMR[BL]. For reads that require less than the full burst length, extraneous data in the burst is ignored and suppressed by the assertion of  $\overline{\text{LSDDQM}}[0:3]$ . For writes that require less than the full burst length, the non-targeted addresses are protected by driving corresponding  $\overline{\text{LSDDQM}}$  bits high (inactive) on the irrelevant cycles of the burst. However, system performance is not compromised because, if a new transaction is pending, the SDRAM controller begins executing it immediately, effectively terminating the burst early.

### 14.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 14-51. In this case, the latched address carries the mode bits for the command.

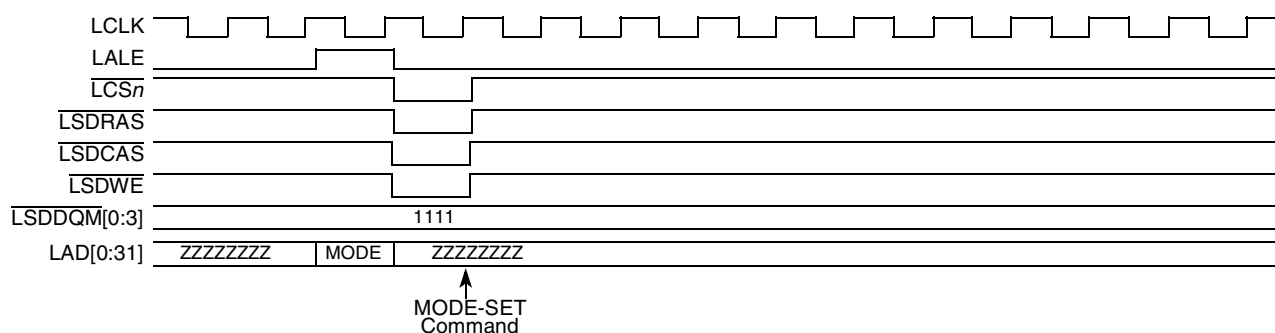


Figure 14-51. SDRAM MODE-SET Command

### 14.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision

between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires; this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

### 14.4.3.11.1 SDRAM Refresh Timing

The SDRAM memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the S

DRAM machine’s mode register (LSDMR[RFCR]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFCR].

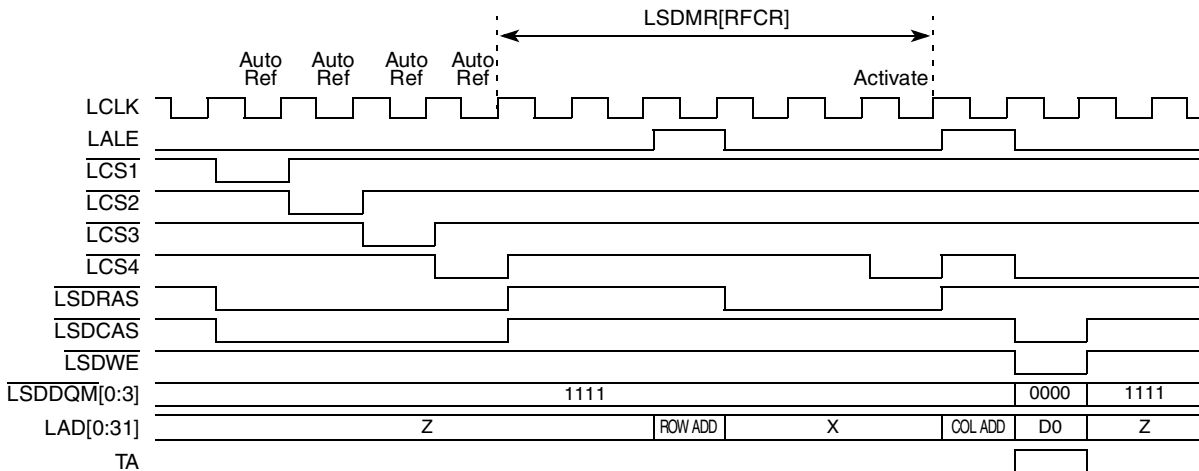


Figure 14-52. SDRAM Bank-Staggered Auto-Refresh Timing

## 14.4.4 User-Programmable Machines (UPMs)

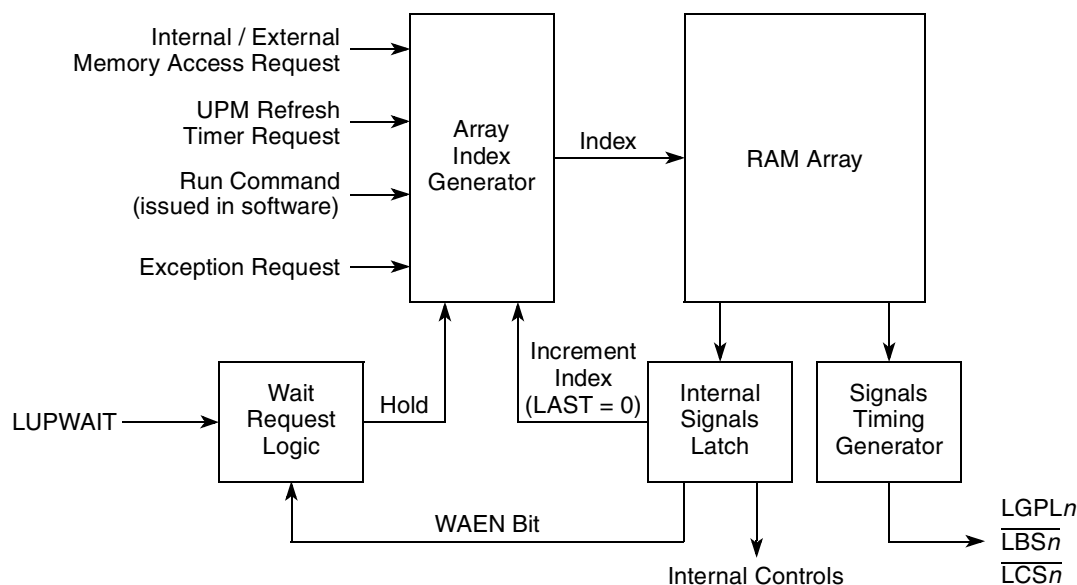
UPMs are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal RAM array that specifies the logical value driven on the external memory control signals ( $\overline{LCSn}$ ,  $\overline{LBS}[0:3]$ , and  $\overline{LGPL}[0:5]$ ) for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock period on the byte select and chip select lines.



**NOTE**

If the  $\overline{\text{LGPL4/LGTA/LUPWAIT/LPBSE}}$  signal is used as both an input and an output, a weak pull-up is required. Refer to the hardware specification for details regarding termination options.

Figure 14-53 shows the basic operation of each UPM.



**Figure 14-53. User-Programmable Machine Functional Block Diagram**

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

#### 14.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each of the possible UPM requests. An internal device's request for a memory access initiates one of the following patterns ( $\text{MxMR}[OP] = 00$ ):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

A UPM refresh timer request pattern initiates a refresh timer pattern (RTS).

An exception (caused by a bus monitor time-out error) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

Figure 14-54 and Table 14-27 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands ( $MxMR[OP] = 11$ ), however, can initiate patterns starting at any of the 64 UPM RAM words.

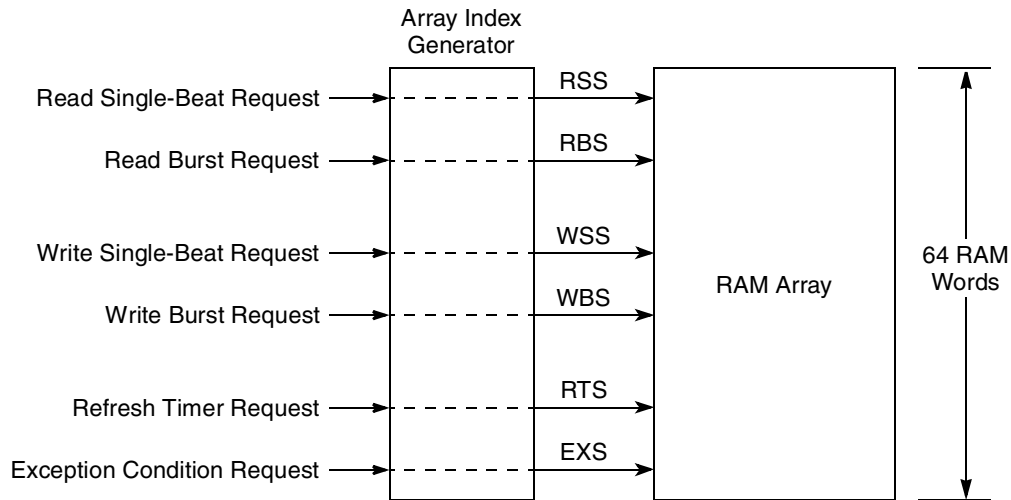


Figure 14-54. RAM Array Indexing

Table 14-27. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

#### 14.4.4.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly 4 double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting  $OR_n[BI]$ . Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

#### 14.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 14-55 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

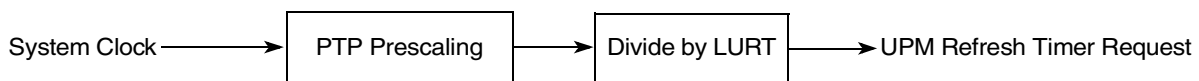


Figure 14-55. Memory Refresh Timer Request Block Diagram

By default, all local bus refreshes are performed using the refresh pattern of UPMA. This means that if refresh is required,  $MAMR[RFEN]$  must be set. It also means that only one refresh routine should be programmed and be placed in UPMA, which serves as the refresh executor. Any banks assigned to a UPM are provided with the refresh pattern if the  $RFEN$  bit of the corresponding UPM is set. UPMA assigned banks, therefore, always receive refresh services when  $MAMR[RFEN]$  is set, while UPMB and UPMC assigned banks also receive (the same) refresh services if the corresponding  $MxMR[RFEN]$  bits are set.

Note that the UPM refresh timer request should not be used in a system with SDRAM refresh enabled. The system designer must choose to use either SDRAM refresh or UPM refresh. Using both may result in missing refresh periods to memory.

#### 14.4.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its  $LAST$  bit set. The RUN command is issued by setting  $MxMR[OP] = 11$  and accessing  $UPM_n$  memory region with any write transaction that hits the corresponding UPM machine.  $MxMR[MAD]$  determines the starting address in the RAM array for the pattern.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

#### 14.4.4.1.4 Exception Requests

When the LBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

#### 14.4.4.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program UPMs:

1. Set up  $BR_n$  and  $OR_n$  registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT and MAMR[RFEN] if refresh is required.
4. Program  $M_xMR$ .

Patterns are written to the RAM array by setting  $M_xMR[OP] = 01$  and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each time followed by a read from MDR and then followed by a (dummy) write transaction to the relevant UPM assigned bank. A read from MDR is required to ensure that the MDR update has occurred prior to the (dummy) write transaction.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when  $M_xMR[OP] = 10$ ).

#### NOTE

$M_xMR$ /MDR registers should not be updated while dummy read/write accesses are still in progress. Dummy transaction completion is indicated by incremented  $M_xMR[MAD]$ . In order to enforce proper ordering between updates to the  $M_xMR$  register and the dummy accesses to the UPM memory region, two rules must be followed:

- 1.) Since the result of any update to the  $M_xMR$ /MDR register must be in effect before the dummy read or write to the UPM region, a write to  $M_xMR$ /MDR should be followed immediately by a read of  $M_xMR$ /MDR.
- 2.) The UPM memory region should have the same MMU settings as the memory region containing the  $M_xMR$  configuration register; both should be mapped by the MMU as cache-inhibited and guarded. This prevents the e500 core from re-ordering a read of the UPM memory around the read of  $M_xMR$ . Once the programming of the UPM array is complete the MMU setting for the associated address range can be set to the proper mode for normal operation, such as cacheable and copyback.

#### 14.4.4.2.1 UPM Programming Example (Two Sequential Writes to the RAM Array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant  $BR_n$  and  $OR_n$  registers have been previously setup.

1. Program  $M_xMR$  for the first write (with desired RAM array address).
2. Write pattern/data to MDR to ensure that the  $M_xMR$  has already been updated with the desired configuration.
3. Read MDR to ensure that the MDR has already been updated with the desired pattern. (Or, read  $M_xMR$  if step 2 is not performed.)
4. Perform a dummy write transaction. (Write transaction can now be performed.)
5. Read/check  $M_xMR[MAD]$ . If incremented, then the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program  $M_xMR$  for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the  $M_xMR$  has already been updated with the desired configuration.
8. Read MDR to ensure that the MDR has already been updated with the desired pattern.
9. Perform a dummy write transaction. (Write transaction can now be performed.)
10. Read/check  $M_xMR[MAD]$ . If incremented, then the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, then step 3 (or 8) is replaced by the following:

- Read  $M_xMR$  to ensure that the  $M_xMR$  has already been updated with the desired configuration.

#### 14.4.4.2.2 UPM Programming Example (Two Sequential Reads from the RAM Array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when  $M_xMR[OP] = 0b10$ ). The following example further illustrates the steps required to perform two reads from the RAM array at non-sequential addresses assuming that the relevant  $BR_n$  and  $OR_n$  registers have been previously setup.

1. Program  $M_xMR$  for the first read with the desired RAM array address.
2. Read  $M_xMR$  to ensure that the  $M_xMR$  has already been updated with the desired configuration, such as RAM array address.
3. Perform a dummy read transaction. (Read transaction can now be performed.)
4. Read/check  $M_xMR[MAD]$ . If incremented, then the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program  $M_xMR$  for the second read with the desired RAM array address.
7. Read  $M_xMR$  to ensure that the  $M_xMR$  has already been updated with the desired configuration, such as RAM array address.
8. Perform a dummy read transaction. (Read transaction can now be performed.)

9. Read/check MxMR[MAD]. If incremented, then the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

### 14.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. Each bit in the RAM word relating to  $\overline{LCS}_n$  and  $\overline{LBS}$  timing specifies the value of the corresponding external signal at each quarter phase of the bus clock.

The division of UPM bus cycles into phases is shown in [Figure 14-56](#).

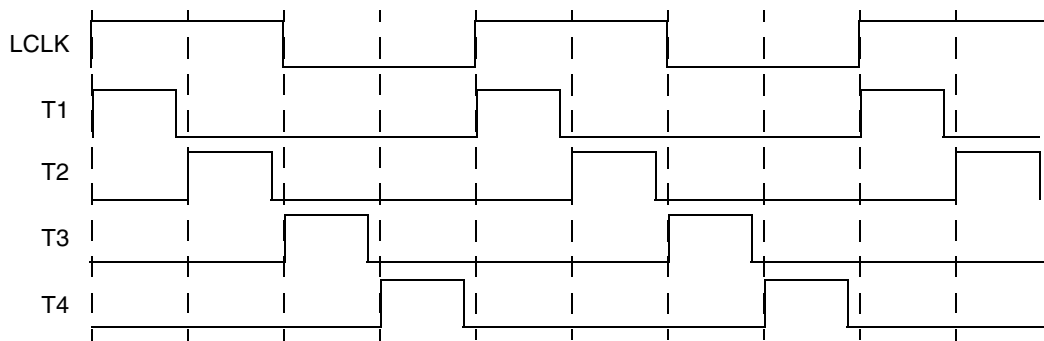


Figure 14-56. UPM Clock Scheme

### 14.4.4.4 RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in [Figure 14-57](#). The signals at the bottom of the figure are UPM outputs. The selected  $\overline{LCS}_n$  is for the bank that matches the current address. The selected  $\overline{LBS}$  is for the byte lanes read or written by the access.

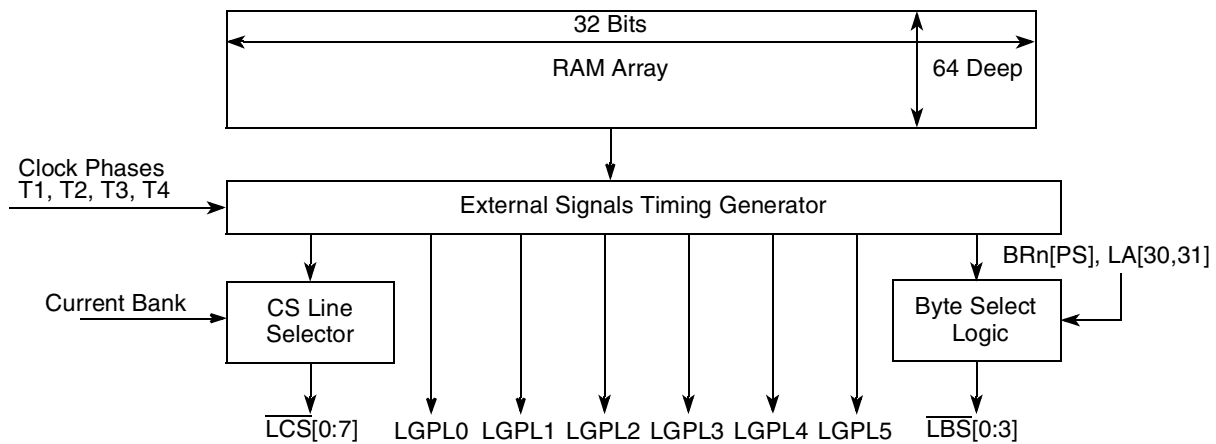
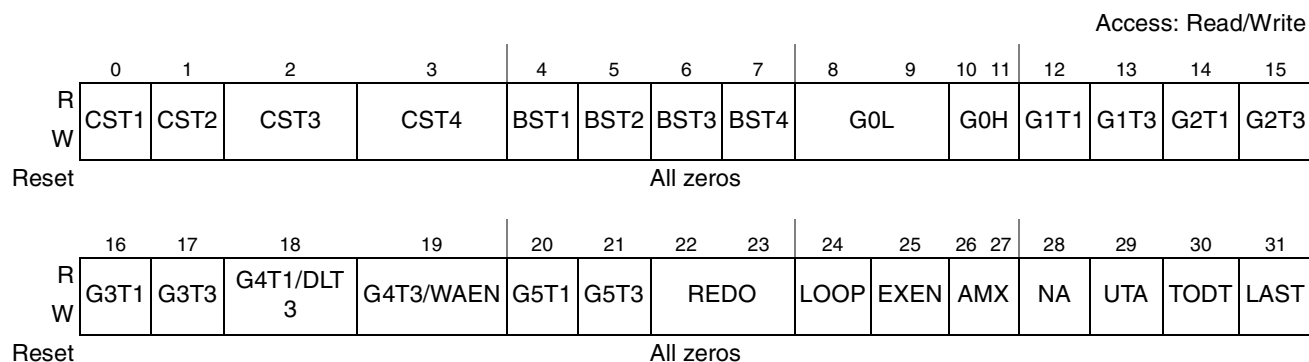


Figure 14-57. RAM Array and Signal Generation

### 14.4.4.4.1 RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 14-58 shows the RAM word fields. The  $CST_n$  and  $BST_n$  bits determine the state of UPM signals  $\overline{LCS}_n$  and  $\overline{LBS}[0:3]$  at each quarter phase of the bus clock.



**Figure 14-58. RAM Word Field Descriptions**

Table 14-28 describes RAM word fields.

**Table 14-28. RAM Word Field Descriptions**

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{LCS}_n$ during bus clock quarter phase 1.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{LCS}_n$ during bus clock quarter phase 2.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of $\overline{LCS}_n$ during bus clock quarter phase 3.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{LCS}_n$ during bus clock quarter phase 4.
4	BST1	Byte select timing 1. Defines the state (0 or 1) of $\overline{LBS}$ during bus clock quarter phase 1.
5	BST2	Byte select timing 2. Defines the state (0 or 1) of $\overline{LBS}$ during bus clock quarter phase 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{LBS}$ during bus clock quarter phase 3.
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{LBS}$ during bus clock quarter phase 4.
8–9	G0L	General-purpose line 0 lower. Defines the state of LGPL0 during the bus clock quarter phases 1 and 2 (first half phase). 00 Value defined by MxMR[G0CL] 01 Reserved 10 0 11 1
10–11	G0H	General-purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by MxMR[G0CL] 01 Reserved 10 0 11 1
12	G1T1	General-purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).

Table 14-28. RAM Word Field Descriptions (continued)

Bits	Name	Description
13	G1T3	General-purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General-purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General-purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General-purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General-purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DLT3	General-purpose line 4 timing 1/delay time 3. The function of this bit is determined by MxMR[GPL4]. If MxMR[GPL4] = 0 and LGPL4/LUPWAIT signal functions as an output (LGPL4), G4T1/DLT3 defines the state (0 or 1) of LGPL4 during bus clock quarter phases 1 and 2 (first half phase). If MxMR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), if a read burst or single read is executed, G4T1/DLT3 defines the sampling of the data bus as follows: 0 In the current word, the data bus should be sampled at the start of bus clock quarter phase 1 of the next bus clock cycle. 1 In the current word, the data bus should be sampled at the start of bus clock quarter phase 3 of the current bus clock cycle.
19	G4T3/WAEN	General-purpose line 4 timing 3/wait enable. Bit function is determined by MxMR[GPL4]. If MxMR[GPL4] = 0 and LGPL4/LUPWAIT signal functions as an output (LGPL4), G4T3/WAEN defines the state (0 or 1) of LGPL4 during bus clock quarter phases 3 and 4 (second half phase). If MxMR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), G4T3/WAEN is used to enable the wait mechanism: 0 LUPWAIT detection is disabled. 1 LUPWAIT is enabled. If LUPWAIT is detected as being asserted, a freeze in the external signals logical values occurs until LUPWAIT is detected as being negated.
20	G5T1	General-purpose line 5 timing 1. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General-purpose line 5 timing 3. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 3 and 4 (second half phase).
22–23	REDO	Redo current RAM word. Defines the number of times to execute the current RAM word. 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	Loop start/end. The first RAM word in the RAM array where LOOP is 1 is recognized as the loop start word. The next RAM word where LOOP is 1 is the loop end word. RAM words between, and including the start and end words, are defined as part of the loop. The number of times the UPM executes this loop is defined in the corresponding loop fields of the MxMR. 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop.



Table 14-28. RAM Word Field Descriptions (continued)

Bits	Name	Description
25	EXEN	Exception enable. Allows branching to an exception pattern at the exception start address (EXS). When an internal bus monitor time-out exception is recognized and EXEN in the RAM word is set, the UPM branches to the special exception start address (EXS) and begins operating as the pattern defined there specifies. The user should provide an exception pattern to negate signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate $\overline{RAS}$ and $\overline{CAS}$ to prevent data corruption. If EXEN = 0, exceptions are ignored by UPM (but not by Local Bus) and execution continues. After the UPM branches to the exception start address, it continues reading until the LAST bit is set in the RAM word. 0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.
26–27	AMX	Address multiplexing. Determines the source of LAD[0:31] during a LALE phase. Any change in the AMX field initiates a new LALE (address) phase. 00 LAD[0:31] is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD[0:31] is the address multiplexed according to MxMR[AM]. For example, row address. 11 LAD[0:31] is the contents of MAR. Used, for example, to initialize a mode. Note that Source ID debug mode is only supported for the AMX = 00 setting.
28	NA	Next burst address. Determines when the address is incremented during a burst access. 0 The address increment function is disabled. 1 The address is incremented in the next cycle. In conjunction with the BRn[PS], the increment value of the state of LA[27:31] is 1, 2 or 4 for port sizes of 8-bits, 16-bits and 32-bits, respectively.
29	UTA	UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle. 0 Transfer acknowledge is not asserted in the current cycle. 1 Transfer acknowledge is asserted in the current cycle.
30	TODT	Turn-on disable timer. The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a $\overline{RAS}$ precharge time. TODT turns the timer on to prevent another UPM access to the same bank until the timer expires. The disable timer period is determined in MxMR[DSn]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored. 0 The disable timer is turned off. 1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.
31	LAST	Last word. When LAST is read in a RAM word, the current UPM pattern terminates and control signal timing set in the RAM word is applied to the current (and last) cycle. However, if the disable timer is activated and the next access is to the same bank, execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in MxMR[DSn]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.

#### 14.4.4.2 Chip-Select Signal Timing (CSTn)

If BRn[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the  $\overline{LCSn}$  for that bank with timing as specified in the UPM RAM word CSTn fields. The selected UPM affects only the assertion and negation of the appropriate  $\overline{LCSn}$  signal. The state of the selected  $\overline{LCSn}$

signal of the corresponding bank depends on the value of each  $CST_n$  bit. Figure 14-59 shows how UPMs control  $\overline{LCS}_n$  signals.

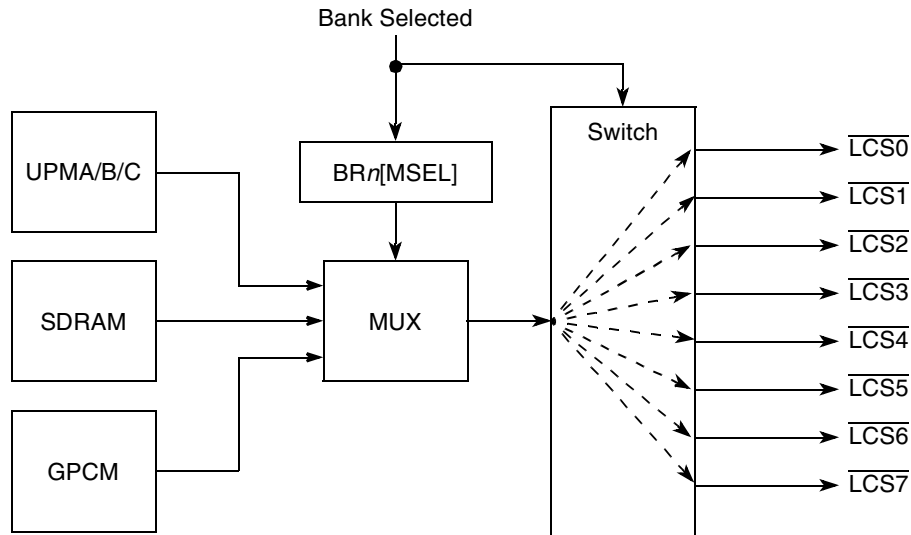


Figure 14-59.  $\overline{LCS}_n$  Signal Selection

#### 14.4.4.4.3 Byte Select Signal Timing (BST $n$ )

If  $BR_n[MSEL]$  of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate  $\overline{LBS}[0:3]$  signal. The timing of all four byte-select signals is specified in the RAM word. However,  $\overline{LBS}[0:3]$  are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

Figure 14-60 shows how UPMs control  $\overline{LBS}[0:3]$ .

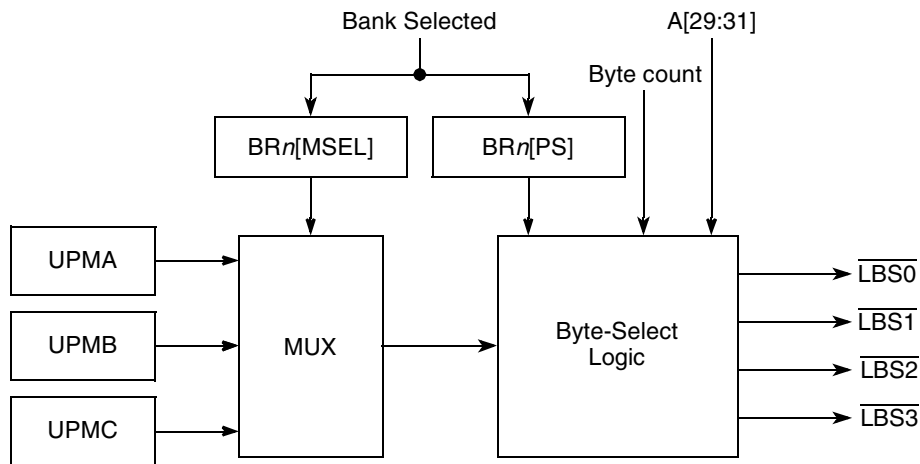


Figure 14-60.  $\overline{LBS}$  Signal Selection

The uppermost byte select ( $\overline{LBS}_0$ ), when asserted, indicates that  $LAD[0:7]$  contains valid data during a cycle. Likewise,  $\overline{LBS}_1$  indicates that  $LAD[8:15]$  contains valid data,  $\overline{LBS}_2$  indicates that  $LAD[16:23]$  contains valid data, and  $\overline{LBS}_3$  indicates that  $LAD[24:31]$  contains valid data. For a UPM refresh timer request, all  $\overline{LBS}[0:3]$  signals are asserted/negated by the UPM according to the refresh pattern only.

Following any internal bus monitor exception,  $\overline{\text{LBS}}[0:3]$  signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

#### 14.4.4.4.4 General-Purpose Signals ( $GnTn$ , $GO_n$ )

The general-purpose signals (LGPL[0:5]) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of the bus clock and/or at the falling edge of the bus clock. LGPL0 offers enhancements beyond the other LGPL $n$  lines.

GPL0 can be controlled by an address line specified in MxMR[G0CL]. To use this feature, G0H and G0L should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

#### 14.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 14-29. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken if LAST and LOOP must not be set together.

**Table 14-29. MxMR Loop Field Use**

Request Served	Loop Field
Read single-beat cycle	RLF
Read burst cycle	RLF
Write single-beat cycle	WLF
Write burst cycle	WLF
Refresh timer expired	TLF
RUN command	RLF

#### 14.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

#### 14.4.4.4.7 Address Multiplexing (AMX)

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between driving the transaction address, driving it according to the multiplexing specified by the M<sub>x</sub>MR[AM] field, or driving the MAR contents on the address signals. In all cases, LA[27:31] of the LBC are driven by the five lsb's of the address selected by AMX, regardless of whether the NA bit of the RAM word is used to increment the current address. The effect of NA = 1 is visible only when AMX = 00 chooses the column address.

Table 14-30 shows how M<sub>x</sub>MR[AM] settings affect address multiplexing when the RAM word AMX = 10. The 16 msbs of the LAD[0:31] bus during an address phase are driven with zero in the AMX = 10 case.

**Table 14-30. UPM Address Multiplexing**

AM	LAD[0:31] as Address Signals	A0–A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external signal when address multiplexing is enabled—RAM word AMX = 10	0	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		0	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		0	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		0	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		0	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Note that any change to the AMX field from one RAM word to the next RAM word executed results in an address phase on the LAD[0:31] bus with the assertion of LALE for the number of cycles set for LALE in the OR<sub>n</sub> and LCRR registers. The LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

#### 14.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the LBC), the value of the DLT3 bit in the same RAM word, in conjunction with  $MxMR[GPLn4DIS]$ , determines when the data input is sampled by the LBC as follows:

- If  $MxMR[GPLn4DIS] = 1$  (G4T4/DLT3 functions as DLT3) and  $DLT3 = 1$  in the RAM word, data is latched on the falling edge of the bus clock instead of the rising edge. The LBC samples the data on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.
- If  $GPLn4DIS = 0$  (G4T4/DLT3 functions as G4T4), or if  $GPLn4DIS = 1$  but  $DLT3 = 0$ , data is latched on the rising edge of the bus clock, which occurs at the end of the current bus clock cycle (normal operation).

Figure 14-61 shows how data sampling is controlled by the UPM.

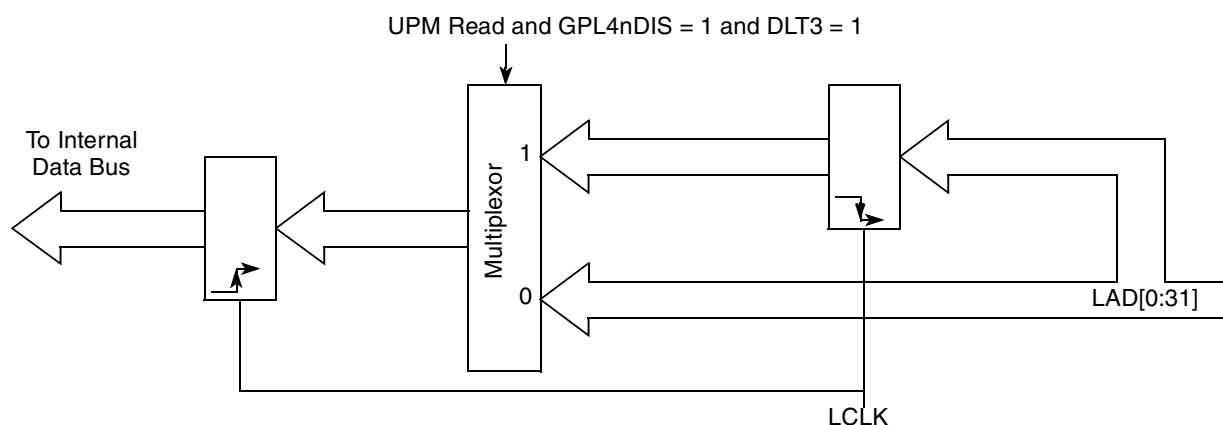


Figure 14-61. UPM Read Access Data Sampling

#### 14.4.4.4.9 LGPL[0:5] Signal Negation (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern is terminated at the end of the current cycle. On the next cycle (following LAST) all the UPM signals are negated unconditionally (driven to logic 1), unless there is a back-to-back UPM request pending. In this case, the signal values for the cycle following the one in which the LAST bit was set are taken from the first RAM word of the pending UPM routine.

#### 14.4.4.4.10 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if  $LAST = 1$  in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and  $WAEN = 1$  in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the

previous RAM word. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 14-62 shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the  $LCSn$  and LGPL1 states and the WAEN value are frozen until LUPWAIT is recognized as negated. WAEN is typically set before the line that contains  $UTA = 1$ . Note that if WAEN and NA are both set in the same RAM word, NA causes the burst address to increment once as normal regardless of whether the UPM freezes.

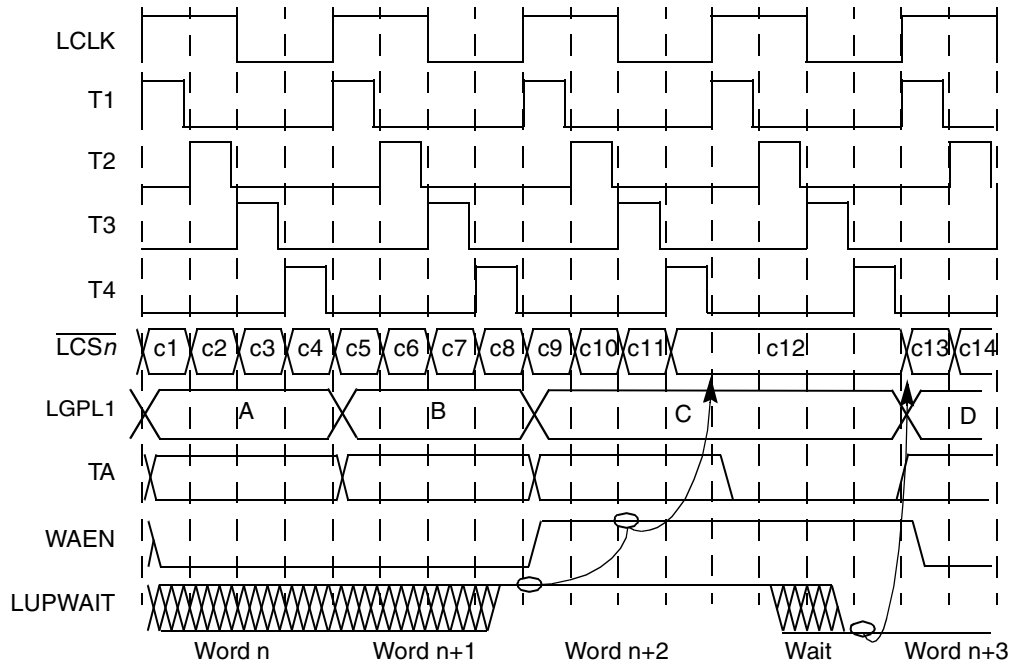


Figure 14-62. Effect of LUPWAIT Signal

#### 14.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both  $WAEN = 1$  and  $UTA = 1$  simultaneously.

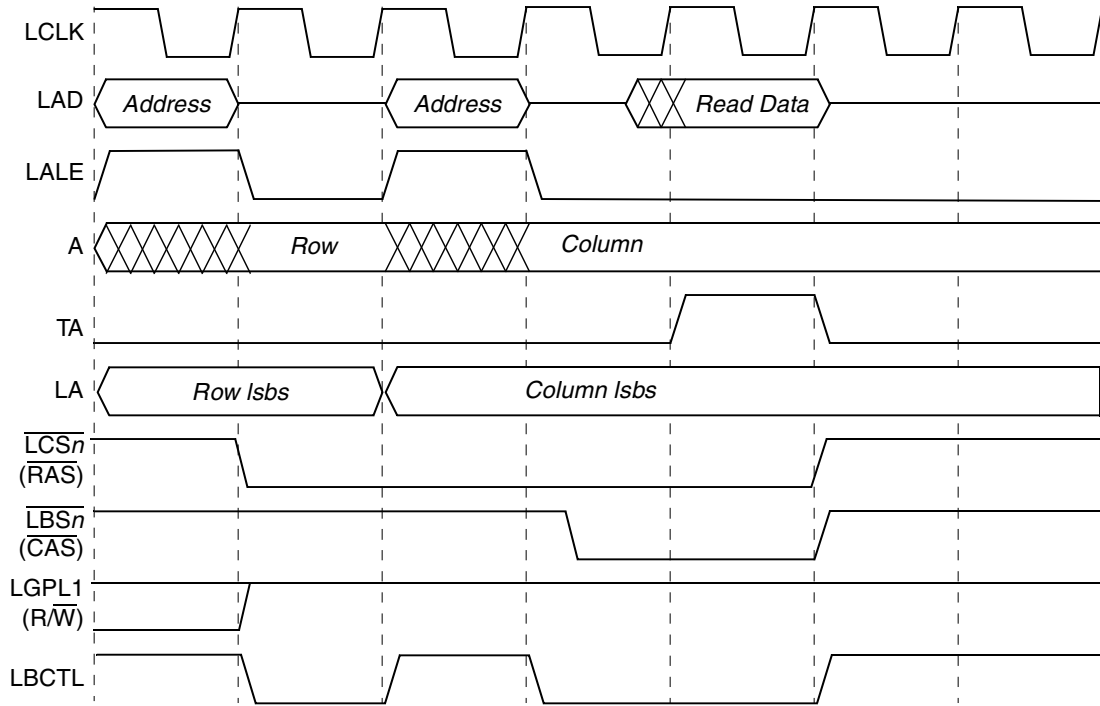
However, programming  $WAEN = 1$  and  $UTA = 1$  in the same RAM word allows UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to affect a data transfer, even if  $UTA$ ,  $WAEN$ , and  $LAST$  are set simultaneously.

#### 14.4.4.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of  $OR_n[TRLX]$  and  $OR_n[EHTR]$ . The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the  $OR_n$  register in addition to any existing bus turn around cycle.

#### 14.4.4.7 Memory System Interface Example Using UPM

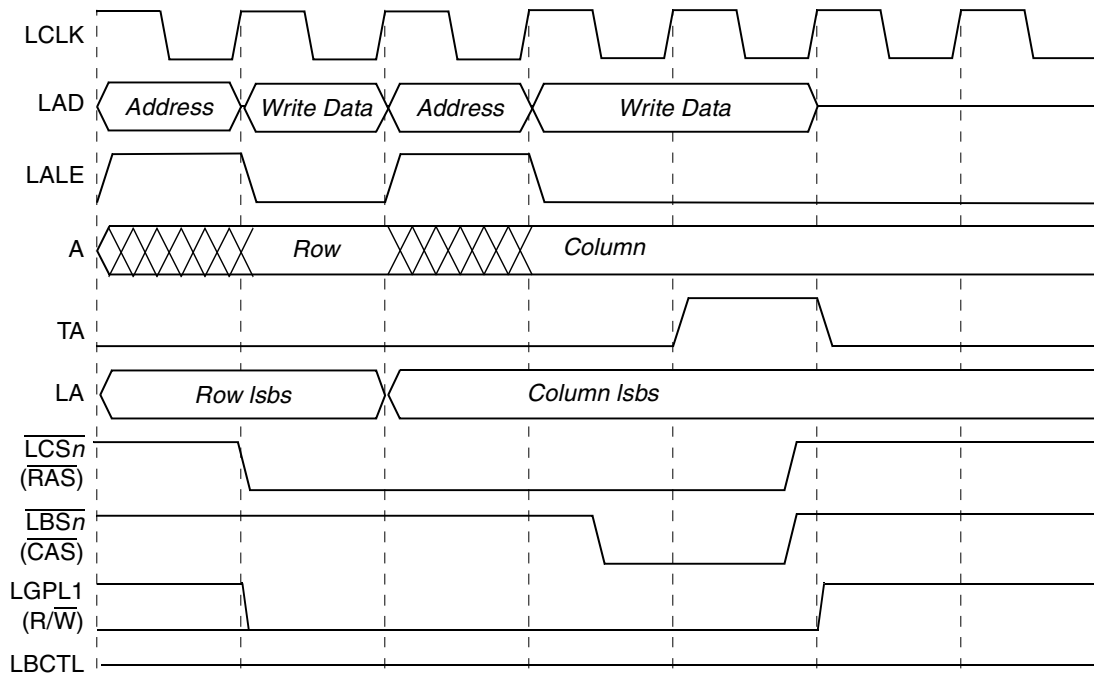
Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section shows timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with  $LCRR[CLKDIV] = 4$  (clock ratio of 8) or 8 (clock ratio of 16). These illustrative examples may not represent the timing necessary for any specific device used with the LBC. Here,  $LGPL1$  is programmed to drive  $R/\overline{W}$  of the DRAM, although any  $LGPL_n$  signal may be used for this purpose.



cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1				Bit 16	
g3t3				Bit 17	
g4t1				Bit 18	
g4t3				Bit 19	
g5t1				Bit 20	
g5t3				Bit 21	
redo[0]				Bit 22	
redo[1]				Bit 23	
loop	0	0	0	Bit 24	
exen	0	0	0	Bit 25	
amx0	1	0	0	Bit 26	
amx1	0	0	0	Bit 27	
na	0	0	0	Bit 28	
uta	0	0	1	Bit 29	
todt	0	0	1	Bit 30	
last	0	0	1	Bit 31	
	RSS	RSS+1	RSS+1	RSS+2	

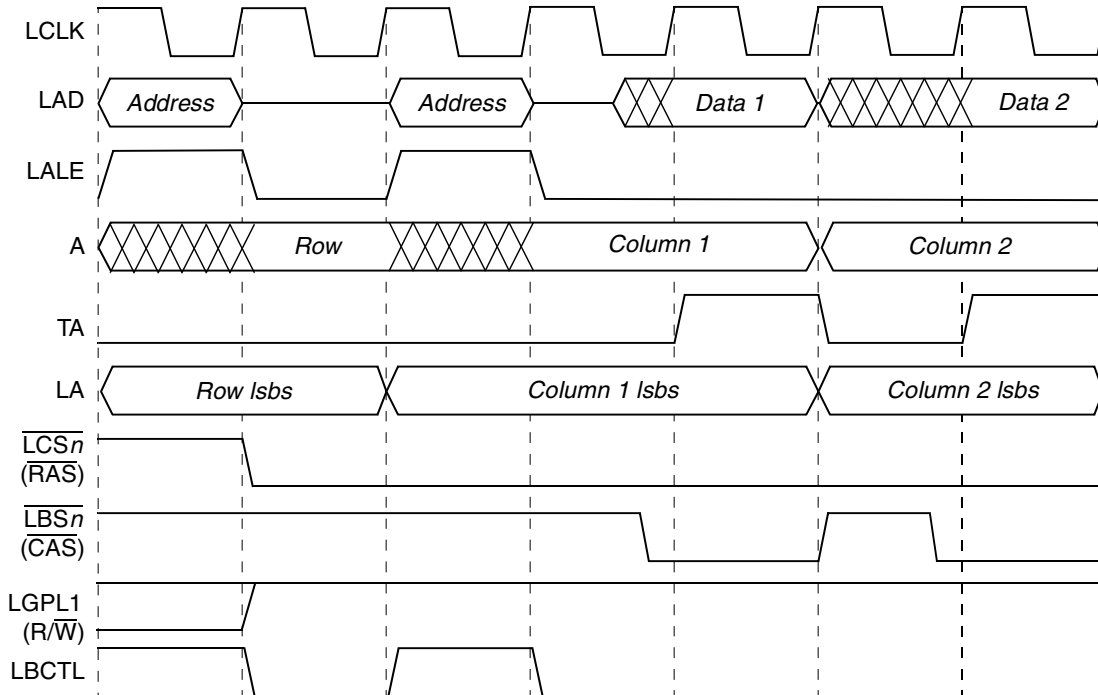
Figure 14-63. Single-Beat Read Access to FPM DRAM





cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	1	Bit 3
bst1	1		1	0	Bit 4
bst2	1		1	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	1	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	0		0	0	Bit 12
g1t3	0		0	0	Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1				Bit 16	
g3t3				Bit 17	
g4t1				Bit 18	
g4t3				Bit 19	
g5t1				Bit 20	
g5t3				Bit 21	
redo[0]				Bit 22	
redo[1]				Bit 23	
loop	0	0	0	Bit 24	
exen	0	0	0	Bit 25	
amx0	1	0	0	Bit 26	
amx1	0	0	0	Bit 27	
na	0	0	0	Bit 28	
uta	0	0	1	Bit 29	
todt	0	0	1	Bit 30	
last	0	0	1	Bit 31	
	WSS	WSS+1	WSS+1	WSS+2	

Figure 14-64. Single-Beat Write Access to FPM DRAM



cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12
g1t3	1		1	1	1	Bit 13
g2t1						Bit 14
g2t3						Bit 15
g3t1						Bit 16
g3t3						Bit 17
g4t1						Bit 18
g4t3						Bit 19
g5t1						Bit 20
g5t3						Bit 21
redo[0]						Bit 22
redo[1]						Bit 23
loop	0		1	1	0	Bit 24
exen	0		0	1	0	Bit 25
amx0	1		0	0	0	Bit 26
amx1	0		0	0	0	Bit 27
na	0		0	1	0	Bit 28
uta	0		0	1	0	Bit 29
todt	0		0	0	1	Bit 30
last	0		0	0	1	Bit 31
	<b>RBS</b>		<b>RBS+1</b>	<b>RBS+2</b>	<b>RBS+3</b>	

Figure 14-65. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)

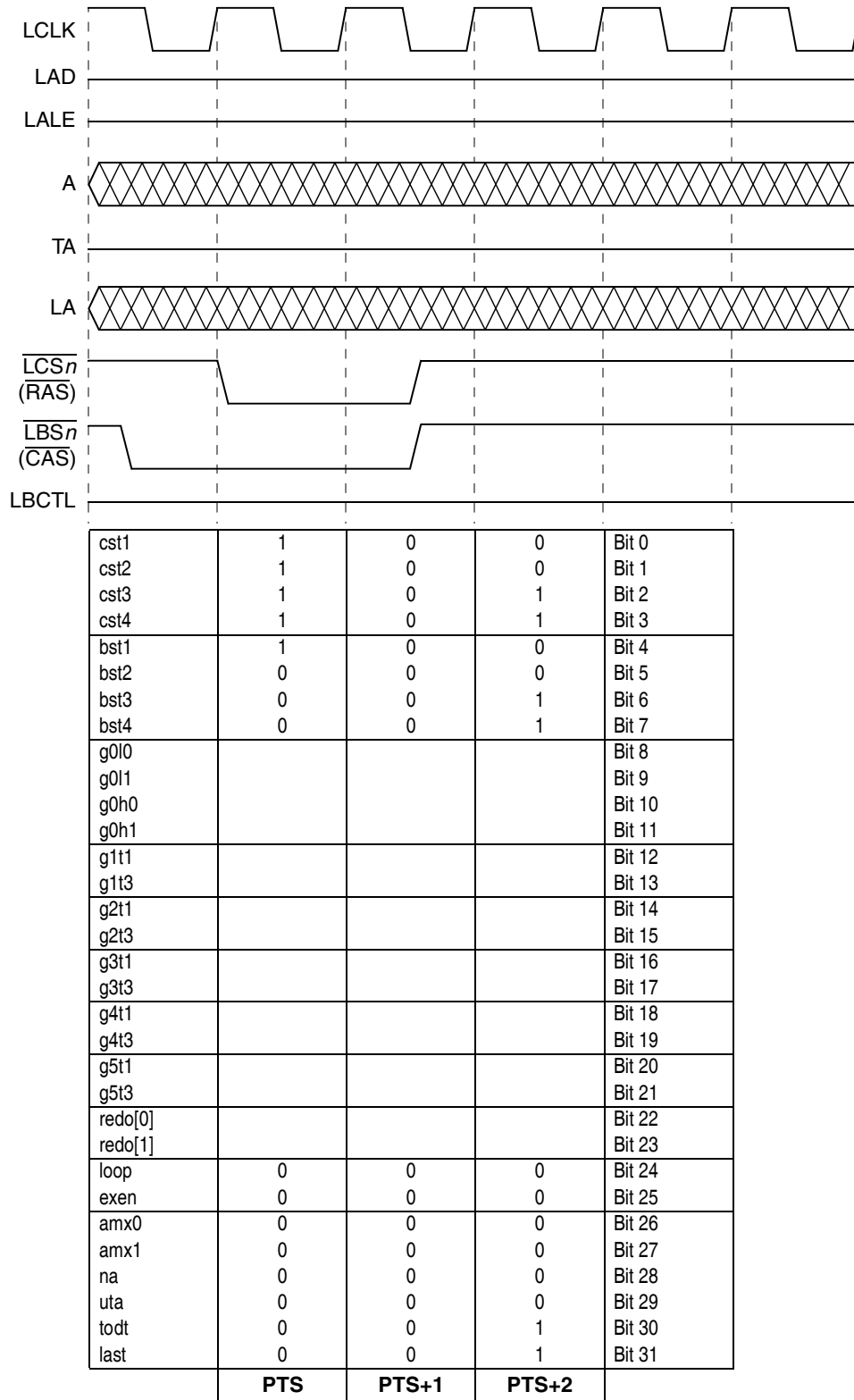


Figure 14-66. Refresh Cycle (CBR) to FPM DRAM

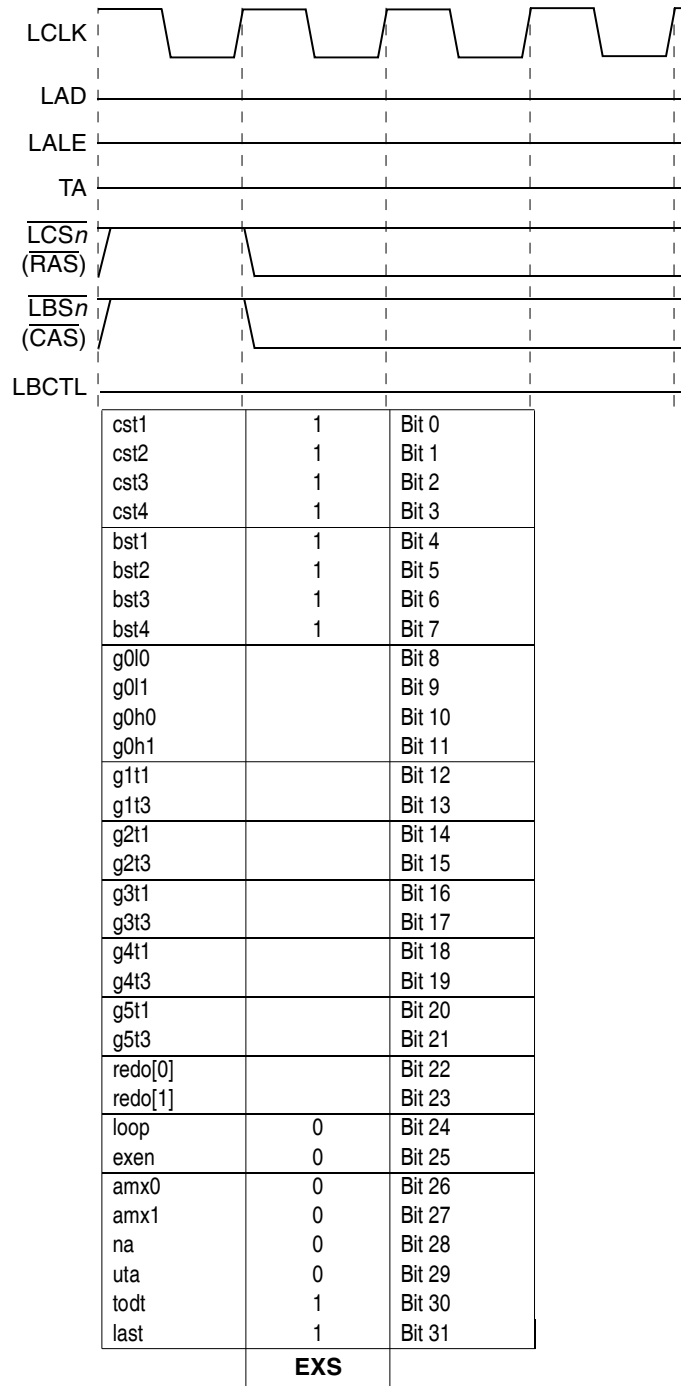


Figure 14-67. Exception Cycle

## 14.5 Initialization/Application Information

### 14.5.1 Interfacing to Peripherals

#### 14.5.1.1 Multiplexed Address/Data Bus and Non-Multiplexed Address Signals

To save signals on the local bus, address and data are multiplexed onto the same 32 bit bus. An external latch is needed to demultiplex and reconstruct the original address. No external intelligence is needed, because the LALE signal provides the correct timing to control a standard logic latch. The LAD signals can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8,16, and 32 bits. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven non-multiplexed. For 8-bit devices, LA[30:31] should be used and for 16-bit devices, LA[30] should be used. 32-bit devices use neither of these signals.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[27:29] are the burst addresses within a natural 32-byte burst. To minimize the amount of address phases needed on the local bus and to optimize the throughput, those signals are driven separately and should be used whenever a device requires the five least significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

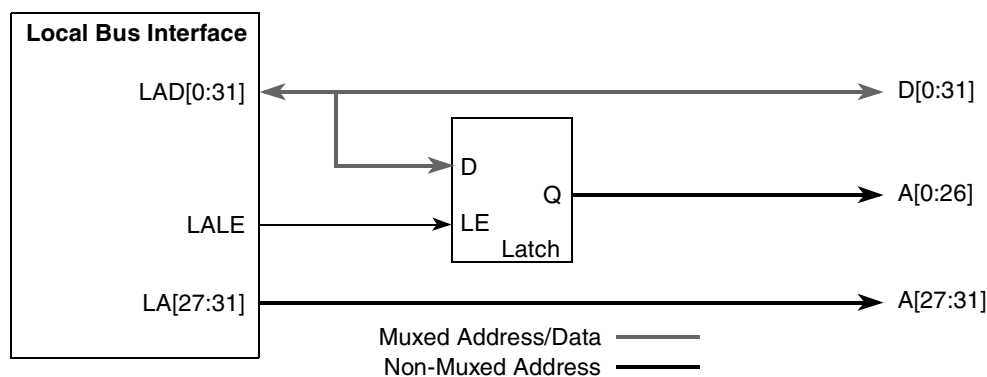


Figure 14-68. Multiplexed Address/Data Bus

### 14.5.1.2 Peripheral Hierarchy on the Local Bus

To achieve high bus speed interfaces for synchronous SRAMs or SDRAMs, a hierarchy of the memories/peripherals connected to the local bus is suggested, as shown in Figure 14-69.

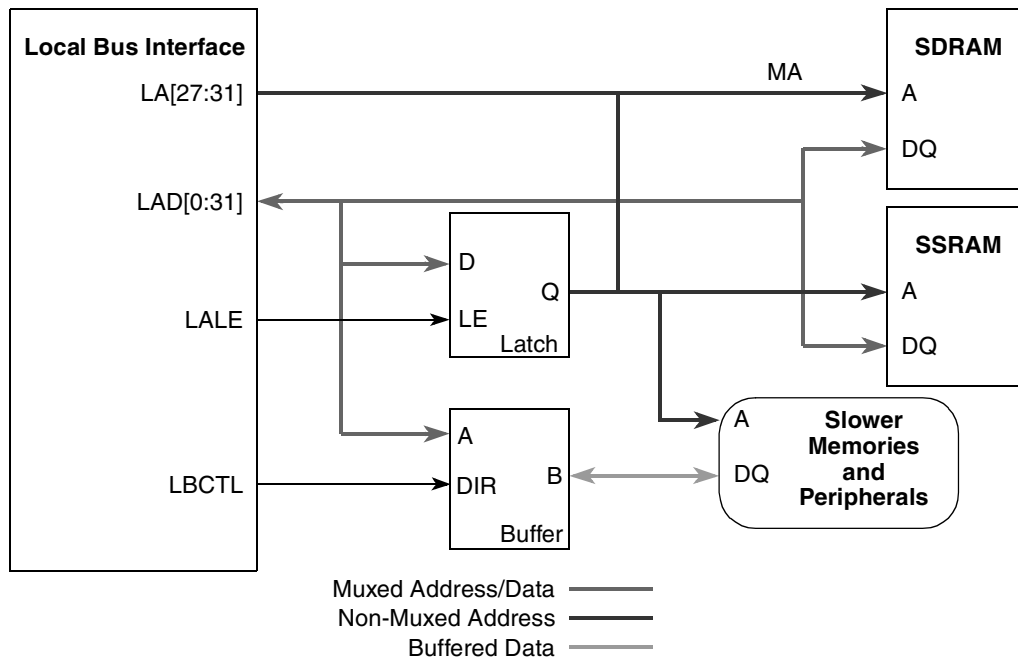


Figure 14-69. Local Bus Peripheral Hierarchy

The multiplexed address/data bus sees the capacitive loading of the data signals of the fast SDRAMs or synchronous SRAMs plus one load for an address latch plus one load for a buffer to the slow memories. The loadings of all other memories and peripherals are hidden behind the buffer and the latch. The system designer needs to investigate the loading scenario and ensure that I/O timings can be met with the loading determined by the connected components.

### 14.5.1.3 Peripheral Hierarchy on the Local Bus for Very High Bus Speeds

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used.

Figure 14-70 shows an example of such a hierarchy. This section is only a guideline and the board designer must simulate the electric characteristics of his scenario to determine the maximum operating frequency.

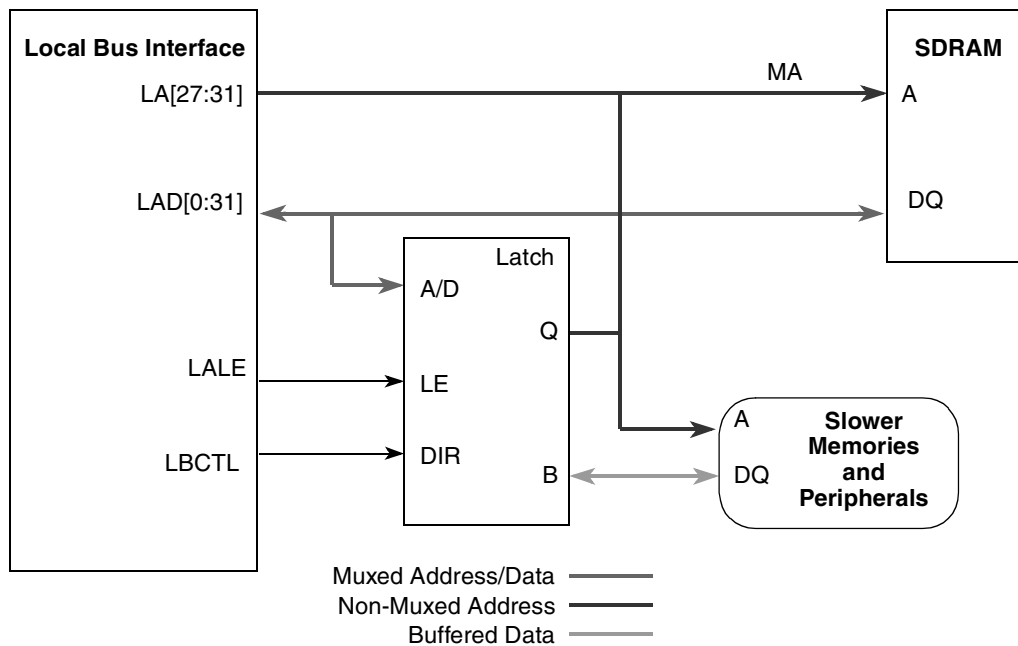


Figure 14-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

#### 14.5.1.4 GPCM Timings

In case a system contains a memory hierarchy with high speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (for example, Flash EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations. Figure 14-71 shows the GPCM address timings.

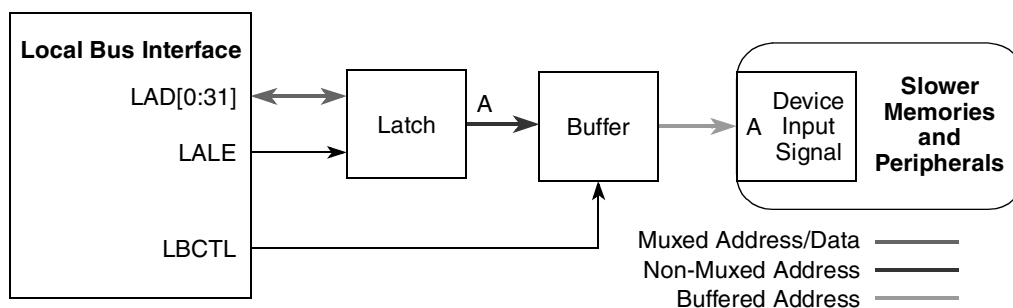


Figure 14-71. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the 2 propagation delays are in the order of 3–6 ns, so for a 166-MHz bus frequency,  $\overline{LCS}$  should arrive on the order of 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered. Figure 14-72 shows the GPCM data timings.

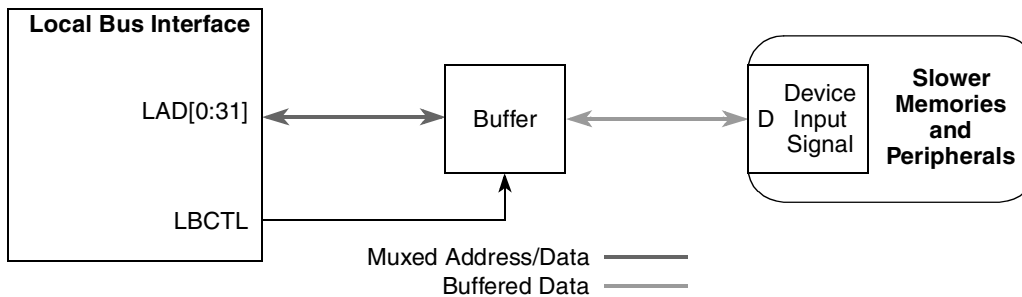


Figure 14-72. GPCM Data Timings

## 14.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

The bus does not change direction for the following cases so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

### 14.5.2.1 Address Phase After Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The LBC then inserts a bus turnaround cycle to avoid contention. The external device has now already placed its data signals in high impedance and no bus contention will occur.

### 14.5.2.2 Read Data Phase After Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its  $t_{dis}(LB)$ . The LBCTL will have its new state after  $t_{en}(LB)$  and, because this is an asynchronous input,



the transceiver starts to drive those signals after its  $t_{en}$ (transceiver) time. The system designer has to ensure, that  $[t_{en}(LB) + t_{en}(\text{transceiver})]$  is larger than  $t_{dis}(LB)$  to avoid bus contention.

### 14.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle will have a new address phase in any case, this basically is the same case as an address phase after a previous read.

### 14.5.2.4 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore, turning around the bus during one pattern. The LBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

## 14.5.3 Interface to Different Port-Size Devices

The LBC supports 8-, 16-, and 32-bit data port sizes. However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on D[0:31], a 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to transfer the maximum amount of data on all bus cycles.

Figure 14-73 shows the device connections on the data bus.

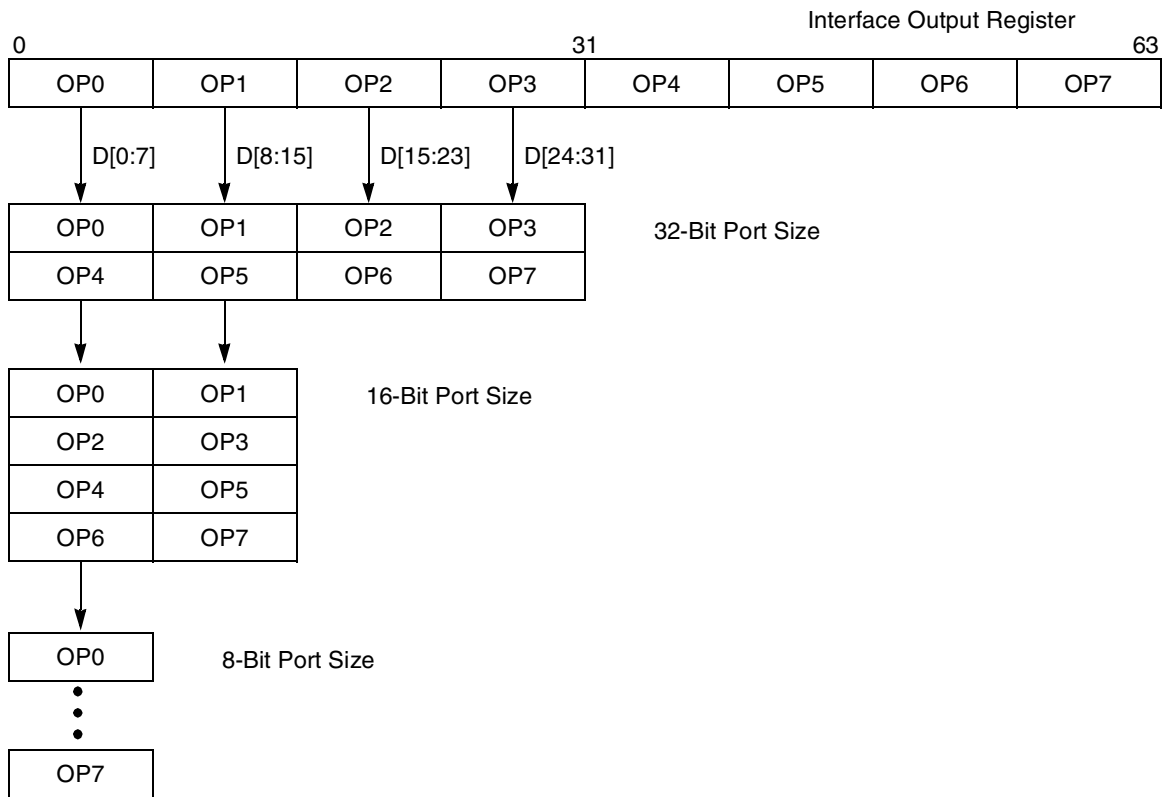


Figure 14-73. Interface to Different Port-Size Devices

Table 14-31 lists the bytes required on the data bus for read cycles.

Table 14-31. Data Bus Requirements For Read Cycle

Transfer Size	Address State <sup>1</sup> A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0-7	8-15	16-23	24-31	0-7	8-15	0-7
Byte	000	OP0 <sup>2</sup>	— <sup>3</sup>	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	OP3	OP3
	100	OP4	—	—	—	OP4	—	OP4
	101	—	OP5	—	—	—	OP5	OP5
	110	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	OP7	—	OP7	OP7

Table 14-31. Data Bus Requirements For Read Cycle (continued)

Transfer Size	Address State <sup>1</sup> A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0-7	8-15	16-23	24-31	0-7	8-15	0-7
Half word	000	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	OP2	OP3	OP2
	100	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	OP6	OP7	OP6	OP7	OP6
Word	000	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	OP4	OP5	OP6	OP7	OP4	OP5	OP4

<sup>1</sup> Address state is the calculated address for port size.

<sup>2</sup> OP $n$ : These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.

<sup>3</sup> — Denotes a byte not driven during that write cycle.

## 14.5.4 Interfacing to SDRAM

The following sections provide application information on interfacing to SDRAM.

### 14.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects ( $\overline{LCSn}$ ) can be programmed to support multiple SDRAM devices. Note that no limitation exists on the number of chip selects that can be programmed for SDRAM. This means that  $\overline{LCS}[1:7]$  can be programmed to support SDRAM, assuming  $\overline{LCS0}$  is reserved for the GPCM to connect to Flash memory.

If multiple chip selects are configured to support SDRAM on the local bus, each SDRAM device should have the same port size and timing parameters. This means that all option registers (OR $n$ ) for the SDRAM chip selects should be programmed exactly the same.

#### NOTE

Although in principle it is possible to mix different port sizes and timing parameters, combinations are limited and this operation is not recommended.

All the chip selects share the same local bus SDRAM mode register (LSDMR) for initialization along with the local bus-assigned SDRAM refresh timer register (LSRT) and the memory refresh timer prescaler register (MPTPR) for refresh.

For refresh, the memory controller supplies auto refresh to SDRAM according to the time interval specified in LSRT and MPTPR as follows:

$$\text{Refresh Period} = \frac{\text{LSRT} \times (\text{MPTPR}[\text{PTP}])}{\text{System Frequency}}$$

This represents the time period required between refreshes. When the refresh timer expires, the memory controller issues a CBR to each chip select. Each CBR is separated by one clock. A refresh timing diagram for multiple chip selects is shown in [Figure 14-52](#) in [Section 14.4.3.11.1](#), “[SDRAM Refresh Timing](#).”

During a memory transaction dispatched to the local bus, the memory controller compares the memory address with the address information of each chip select (programmed with  $BR_n$  and  $OR_n$ ). If the comparison matches a chip select that is controlled by SDRAM, the memory controller requests service to the local bus SDRAM machine, depending on the information in  $BR_n$ . Although multiple chip selects may be programmed for SDRAM, only one chip select is active at any given time; thus, multiple chip selects can share the same SDRAM machine.

#### 14.5.4.2 Maximum Amount of SDRAM Supported

[Table 14-32](#) summarizes information based on typical SDRAM data sheets.

**Table 14-32. Typical SDRAM Devices**

SDRAM Device	64-Mbit				128-Mbit				256-Mbit				512-Mbit			
	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
I/O Port	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Bank	12	12	12	11	12	12	12	12	13	13	13	13	13	13	13	—
Row	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	—
Column																

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires 4 SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, 1 chip select provides 128-Mbit/device  $\times$  4 devices = 64 Mbytes. If 4 chip selects are programmed for SDRAM use, the result is 64 Mbytes  $\times$  4 = 256 Mbytes. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbytes. Consequently, 512-Mbit devices allow for 1 Gbyte.

Although there is no technical difficulty in supporting multiple chip select configurations, in practice, the user may want to maximize the amount of SDRAM assigned to each chip select to minimize cost.

### 14.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

#### 14.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select Multiplexing

LSDMR[BSMA] is used to multiplex the bank select address. The BSMA field and corresponding multiplexed address are shown below:

```

000 LA12-LA13
001 LA13-LA14
...
111 LA19-LA20

```

Note that LA12 is the latched value of LAD12.

The highest address signals that the bank selects can be multiplexed with are LA[12:13], which limits the signals for the row address to LA[14:31]. For a 32-bit port, the maximum width of the local bus, LA[30:31] are not connected, and the maximum row is LA[14:29]. The local bus SDRAM machine supports 15 rows, which is sufficient for all devices.

#### 14.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address signals to leave room for future upgrades. For example, a user could multiplex the bank select signals to LA[14:15], leaving LA16 to connect to the address signal for a larger memory size.

This allows the system designer to design one board that can be used with a current generation of SDRAM devices and upgraded to the next generation without requiring a new board layout.

### 14.5.4.3.3 128-Mbyte SDRAM

Figure 14-74 shows the connection to an SDRAM of 128 Mbytes. Note that all circuit diagrams are principal connection diagrams and do not show any means of signal integrity.

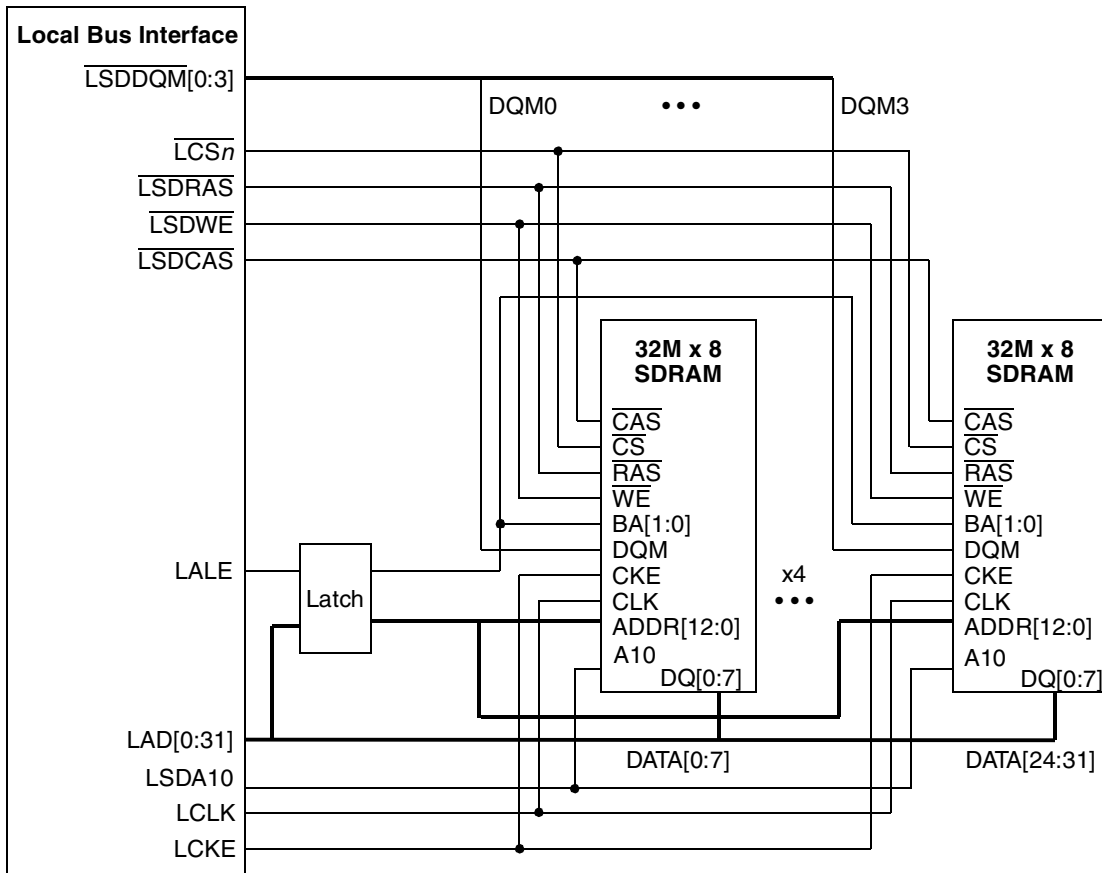


Figure 14-74. 128-Mbyte SDRAM Diagram

Table 14-33 shows details about LAD<sub>n</sub> signal connections for the example in Figure 14-74.

Table 14-33. LAD<sub>n</sub> Signal Connections to 128-Mbyte SDRAM

LAD (Latch Address)	SDRAM Address Signal
LAD29	A0
LAD28	A1
LAD27	A2
LAD26	A3
LAD25	A4
LAD24	A5
LAD23	A6
LAD22	A7

**Table 14-33. LAD<sub>n</sub> Signal Connections to 128-Mbyte SDRAM (continued)**

LAD (Latch Address)	SDRAM Address Signal
LAD21	A8
LAD20	A9
LAD19 (no connect)	A10 is connected to LSDA10
LAD18	A11
LAD17	A12
LAD16	BA0 (if LSDMR[BSMA] = 011)
LAD15	BA1 (if LSDMR[BSMA] = 011)

Consider the following SDRAM organization:

- The 32-bit port size is organized as  $8 \times 8 \times 32$  Mbits.
- Each device has 4 internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in [Table 14-34](#).

**Table 14-34. Logical Address Bus Partitioning**

A[0:4]	A[5:17]	A[18:19]	A[20:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters are extracted:

- COLS = 011, 10 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as shown in [Table 14-35](#).

**Table 14-35. SDRAM Device Address Port During Address Phase**

LA[0:14]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select A[18:19]	Row A[5:17]	No connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

[Table 14-36](#) shows the address port configuration during a READ/WRITE command.

**Table 14-36. SDRAM Device Address Port During READ/WRITE Command**

LA[0:14]	LA[15:16]	LA[17:18]	LA[19]	LA[20:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	No connect

Table 14-37 shows the register configuration for this example. PSRT and MPTPR are not shown but should be programmed according to the device's specific refresh requirements.

**Table 14-37. Register Settings for 128-Mbyte SDRAMs**

Register	Field	Value
BR $n$	BA	Base address
	XBA	Ext. Base Address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR $n$	AM	11_1111_1000_0000_0000_0
	XAM	11
	COLS	011
	ROWS	100
LSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

#### 14.5.4.3.4 256-Mbyte SDRAM

This example uses the same Micron SDRAM as in the previous example, but doubles the number of devices connected and therefore uses two chip selects.

#### 14.5.4.3.5 512-Mbyte SDRAM

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is  $8 \times 4 \times 64$  Mbit  $\times$  2 chip select lines.
- Each device has 4 internal banks, 13 row address lines, and 11 column address lines.

The logical address is partitioned as shown in Table 14-38.

**Table 14-38. Logical Address Partitioning**

A[0:3]	A[4:16]	A[17:18]	A[19:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- COLS = 100, 11 column lines
- ROWS = 100, 13 row lines



During the address phase, the SDRAM address port is set as in [Table 14-39](#).

**Table 14-39. SDRAM Device Address Port During Address Phase**

LA[0:13]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select (A[17:18])	Row (A[4:16])	No-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

[Table 14-40](#) shows the address port settings during a READ/WRITE command.

**Table 14-40. SDRAM Device Address Port During READ/WRITE Command**

LA[0:14]	LA[15:16]	LA[17]	LA[18]	LA[19:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	No-connect

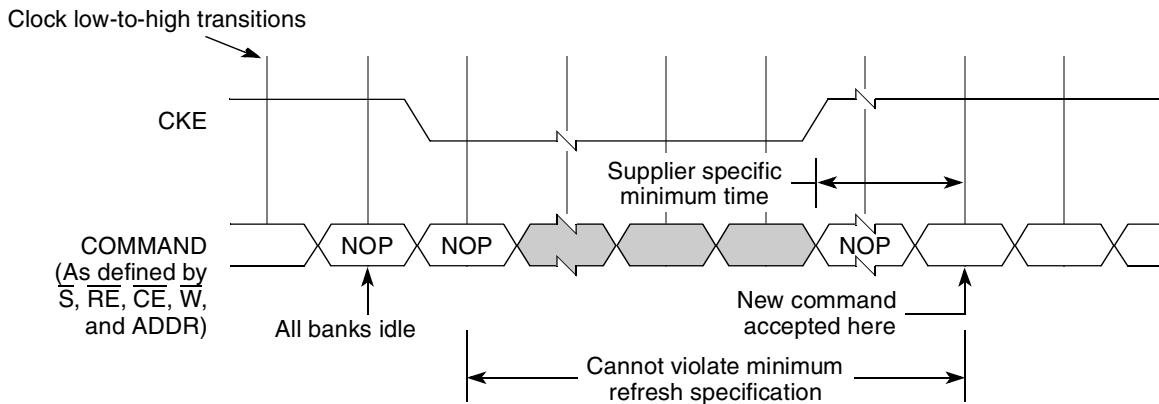
[Table 14-41](#) shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

**Table 14-41. Register Settings for 512-Mbyte SDRAMs**

Register	Field	Value
BR <sub>n</sub>	BA	Base address
	XBA	ext. Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR <sub>n</sub>	AM	11_1110_0000_0000_0000_0
	XAM	11
	BPD	01
	COLS	100
	ROWS	100
PSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

#### 14.5.4.3.6 Power-Down Mode

SDRAMs offer a power-down mode during which the device is not refreshed, and therefore, data is not maintained. This mode is invoked by driving CKE low, while all internal banks are idle; note that they must be precharged first. [Figure 14-75](#) shows the timing. Note that the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

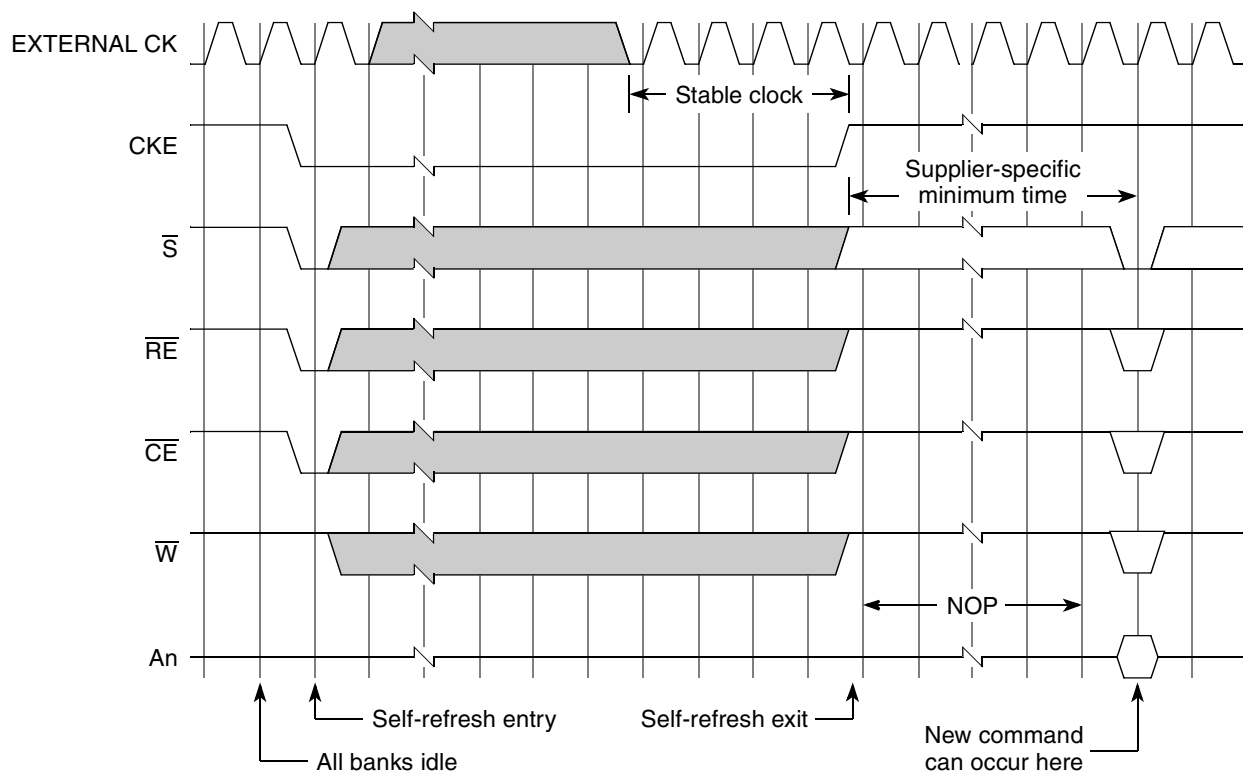


**Figure 14-75. SDRAM Power-Down Timing**

CKE remains low, as long as the device is powered down. After CKE transitions to high, the SDRAM exits the power-down mode.

#### 14.5.4.3.7 Self-Refresh

In order to be able to stop activity on the local bus (for power save or debug), while the content of the SDRAM is maintained, the self-refresh mode is supported. This mode is invoked by issuing a self-refresh command to the SDRAM. The LBC applies the same timing as for the auto refresh, but also pulls the SDRAM CKE (LCKE) signal low in the same cycle. This can only be done with all banks being idle; the SDRAM machine must precharge them ahead of this. As long as CKE stays low, the device refreshes itself and does not need to see any refreshes from the local bus. To exit self refresh, CKE simply has to be pulled high. Note that after returning from self-refresh mode the SDRAM needs a supplier-specific time before it can accept new commands and the auto-refresh mechanism has to be started again. [Figure 14-76](#) shows this timing. The SDRAM controller always uses 200 local bus clocks, which should satisfy any SDRAM requirements. As in the case of the power-down mode, the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command. Refer to [Section 14.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,”](#) for SDRAM interface commands and information on the self-refresh command.



**Figure 14-76. SDRAM Self-Refresh Mode Timing**

#### 14.5.4.3.8 SDRAM Timing

To allow for very high speeds on the memory bus, the capacitive loading on the local bus must be taken into consideration as shown in [Table 14-42](#).

**Table 14-42. SDRAM Capacitance**

Signal	Min	Max	Unit
CLK	2.0	4.0	pf
$\overline{RAS}$ , $\overline{CAS}$ , $\overline{WE}$ , $\overline{CS}$ , CKE, DQM	2.0	5.0	pf
Address	2.0	5.0	pf
DQ <sub>0</sub> –DQ <sub>31</sub>	3.5	6.5	pf

**Note:** Capacitance values compiled from worst case numbers from various data sheets from Samsung and Micron

To implement a system using the hierarchy described earlier for 2 synchronous memory banks, 1 address latch, and 1 buffer loading the multiplexed address/data bus sees a loading of 4 loads of about 6.5 pF maximum. For a nominal load, 30 pF can be used.

Table 14-43. SDRAM AC Characteristics

Parameter		Device Speed				Unit
		166 MHz		133 MHz		
		Min	Max	Min	Max	
CLK cycle time	CAS latency = 3	6	1000	7.5	1000	ns
	CAS latency = 2	—		7.5		
CLK to valid output delay	CAS latency = 3	—	5	—	5.4	ns
	CAS latency = 2	—	—	—	5.4	
Output data hold time	CAS latency = 3	2.5	—	3	—	ns
	CAS latency = 2	—	—	3	—	
Input setup time		1.5	—	2	—	ns
Input hold time		1	—	1	—	ns
CLK to output in Hi-Z	CAS latency = 3	—	5	5.4	—	ns
	CAS latency = 2	—	—	5.4	—	

**Note:** AC characteristics compiled from worst-case numbers from various data sheets from Samsung and Micron.

### Setup and Hold Timing Calculations:

Address TOF (time of flight): board layout delay

Data TOF (time of flight): board layout delay

Clock skew (time of flight): clock skew between the LBC and the clock at the memory device. The local bus PLL feedback mechanism must be used to control this skew to optimize the timing margins, as described in the rest of this subsection.

Address setup margin = cycle time – local bus address CTQ – SDRAM address input setup time – address TOF + clock skew

Address hold margin = local bus address output hold time + address TOF – SDRAM address input hold time – clock skew

Data write to SDRAM setup margin = cycle time – local bus data CTQ – SDRAM data input setup time – data TOF + clock skew

Data write to SDRAM hold margin = local bus data output hold time + data TOF – SDRAM data input hold time – clock skew

Data read from SDRAM setup margin = cycle time – SDRAM data CTQ – local bus data input setup time – data TOF – clock skew

Data read from SDRAM hold margin = SDRAM data output hold time + data TOF – local bus data input hold time + clock skew

To improve the timing margins a PLL is used to generate external clocks, which minimize the skew between the local bus and the memory clock. Figure 14-77 shows relative timings for the local bus clock PLL.

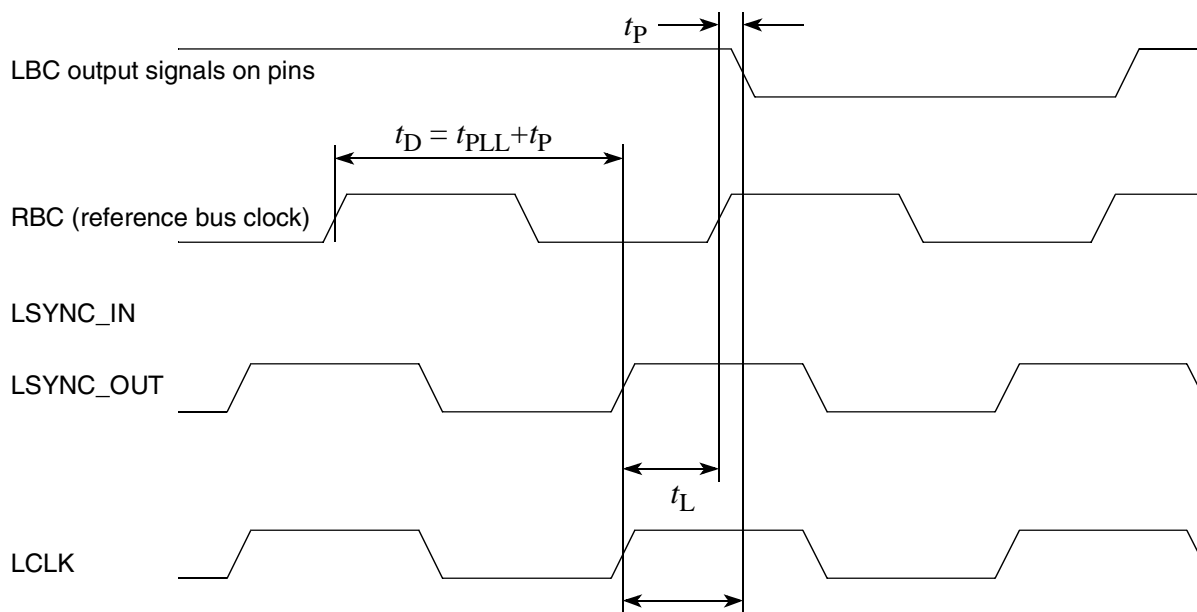


Figure 14-77. Local Bus PLL Operation

#### 14.5.4.4 Parity Support for SDRAM

Contrary to older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16, or x32. There are no mainstream devices that include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Since the local bus allows for SDRAM accesses with less than the full port size, read-modify-write cycles are supported for SDRAM write cycles.

Figure 14-78 shows a connection diagram.

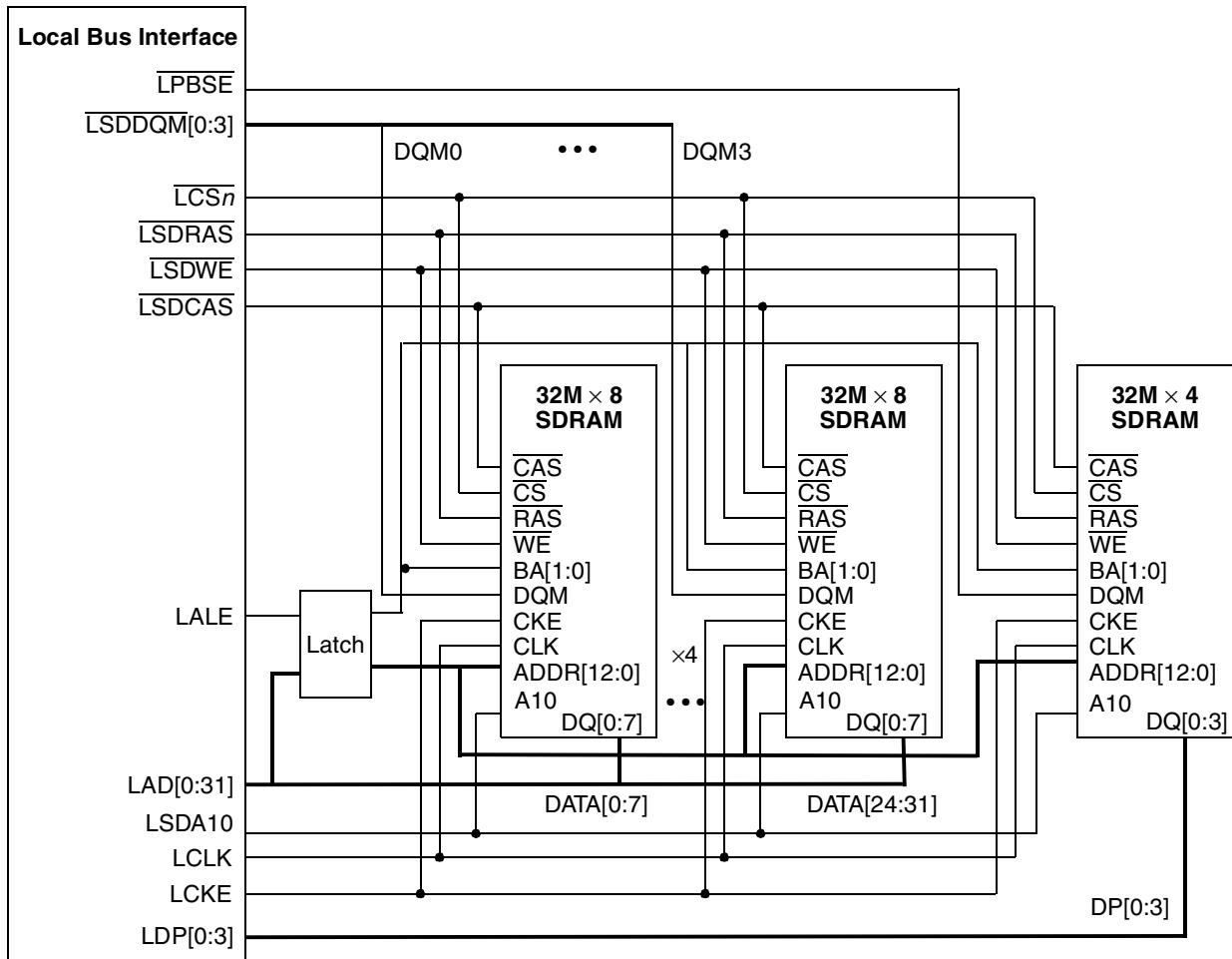


Figure 14-78. Parity Support for SDRAM

### 14.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, memory access patterns are often random and SDRAM is not optimized for that case. ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 14-79 shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs will mostly be used by performance-critical applications, we assume here that, typically, the maximum width of the local bus of 32 bits will be used.

ZBT SRAMs allow different configurations. For the local bus the burst order should be set to linear burst order by tying the mode signal to GND;  $\overline{CKE}$  should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates eight-beat transactions (for 32-bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the LBC generates the new A27 for the second burst.

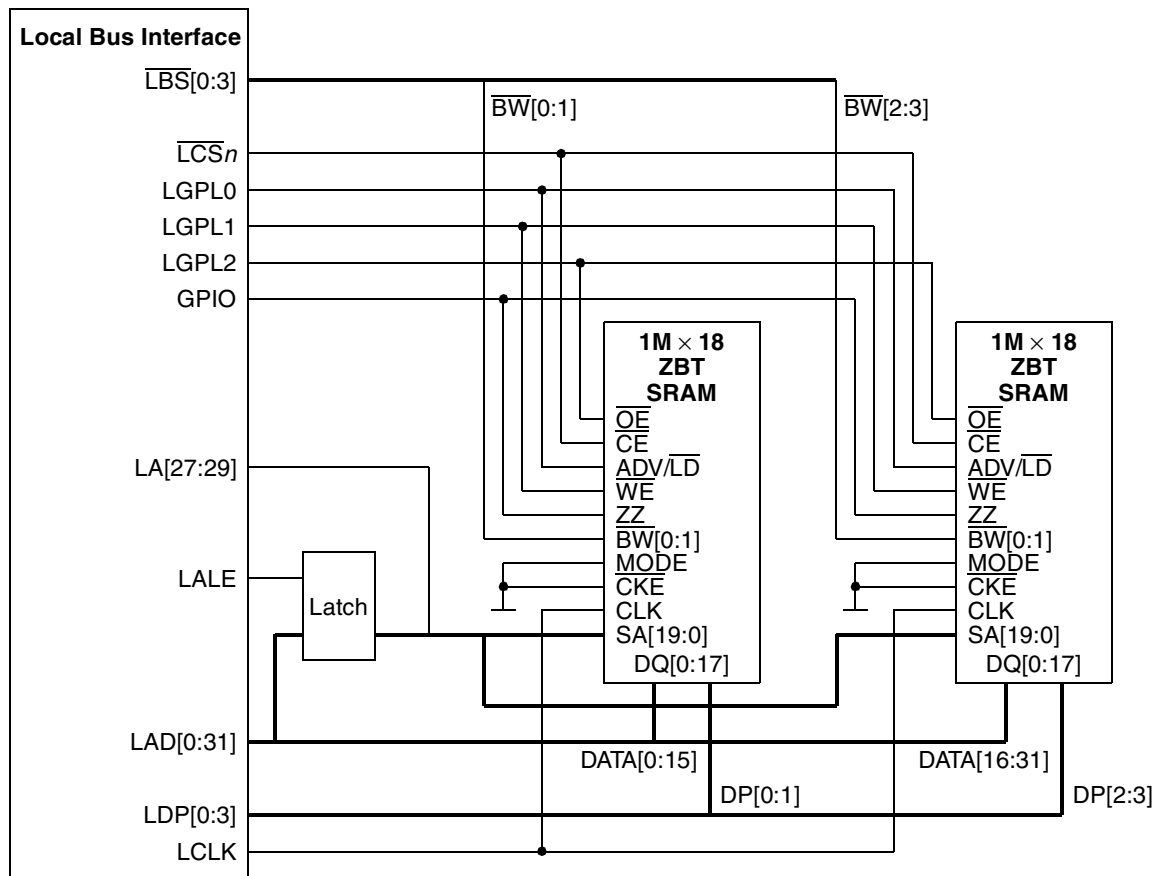


Figure 14-79. Interface to ZBT SRAM

Because we use linear burst on the SRAM, the device will itself burst with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7]. Therefore, two consecutive linear bursts of the ZBT SRAM with A27 = 0 for the first burst and A27 = 1 for the second burst give the desired burst pattern.

The UPM also supports single beat accesses. Because the ZBT SRAM does not support this and always responds with a burst, the UPM pattern has to take care that data for the critical beat is provided (for write) or sampled (for read), and that the rest of the burst is ignored (by negating  $\overline{WE}$ ). The UPM controller basically has to wait for the end of the SRAM burst to avoid bus contention with further bus activities.

ZBT SRAMs have a power down mode, which is invoked by the ZZ signal. Connecting a GPIO signal to ZZ allows use of that power down mode; however, accesses to the SRAM while in power down mode do not create valid results. This should be taken care of by the system software.

Another observation is that SRAMs are available with natural parity. In the example, we use a  $\times 18$  SRAM, which holds two data bytes and two parity bits. While for the support of parity on SDRAM banks the local

bus has to use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

## 14.5.6 Interfacing to DSP Host Ports

In many applications, an integrated communications processor aggregates traffic for DSPs and distributes that traffic to the DSPs. The local bus allows connection to a variety of different DSP host ports and this section gives some information on how to interface to some example DSPs.

### 14.5.6.1 Interfacing to MSC8101 HDI16

This section describes how to interface to the HDI16 peripheral interface of the MSC8101. After initial set-up of the interface, the host and HDI16 device can communicate either by a read and write transaction from the core or, if the setup on the DSP and the host are implemented appropriately, by DMA transfers of the host DMA controller, which can be triggered automatically by signals generated by the HDI16 peripheral.

#### 14.5.6.1.1 HDI16 Peripherals

The host interface (HDI16) is a 16-bit-wide, full-duplex, double-buffered parallel port that can directly connect to the data bus of a host processor. It supports a variety of buses and gluelessly connects with a number of industry-standard microcomputers, microprocessors, and DSPs. The HDI16 also supports the 8-bit host data bus, which makes it fully compatible with the DSP56300 HI08 (as viewed by the host side, not from the DSP side).

The host bus can operate asynchronously to the SC140 core clock, and the HDI16 registers are divided into two banks. The host register bank is accessible to the external host, and the core register bank is accessible to the SC140 core.

The MSC8101 HDI16 host port peripheral has two sets of 16-bit-wide registers—one set is only visible internally to the DSP, while the other set is visible only to the external host processor. [Figure 14-81](#) illustrates the relationship between the two sides.

All of the HDI16 peripheral's registers are mapped directly onto the MSC8101 QBus, as defined by the *MSC8101 16-Bit Digital Signal Processor Reference Manual* (MSC8101RM); the transmit and receive FIFOs are mapped onto the DMA data bus such that the DMA controller can access them directly without core intervention. The addressing for each of these registers is defined in [Section 14.5.6.1.2, "Physical Interconnections."](#)

The HDI16 host port itself is a 16-bit-wide parallel port with various strobe and multiplexing options.

The most important HDI16 host port facet is that it is specified as an asynchronous interface and so reduces concerns over clock skew between the HDI16 host port and the host device's buses. Furthermore, with all the host port registers being accessed with a single chip select and four address lines, as far as the local bus is concerned, the DSP host port is akin to an asynchronous memory mapped region. So, for the HDI16 port in single strobe mode, the host device asserts a chip select, a single data strobe and a read/write line to select HDI16 read or write bus operations.



The read and write strobes are also used as the data latch control to complete the bus transactions, obviating the need for any handshake termination signal from the DSP. The UPM programmer is responsible for satisfying the AC timings of the HDI16 transactions.

Through appropriate mode selection, the HDI16 peripheral's feature set can be fully supported by the local bus's UPM controlled signals. The UPM defined interface can be used with any of the local bus's eight chip selects to give the 16-bit port size and strobe generation that matches that of the HDI16 host port.

Figure 14-80 shows the internal register diagram of the HDI16.

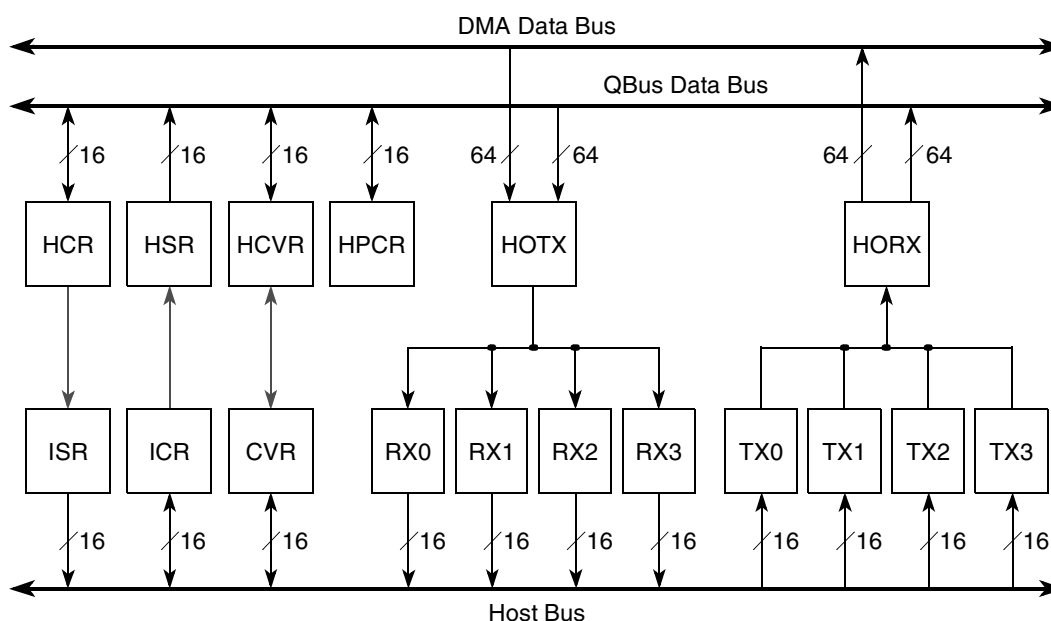


Figure 14-80. MSC8101 HDI16 Peripheral Registers

#### 14.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 14-44 and Figure 14-81.

Table 14-44. Local Bus to MSC8101 HDI16 Connections

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
HD[0:15]	I/O/Z	Host data bus	LAD[0:15]
HA[0]	I	Host address line	Either LA[27] or latched A25
HA[1]	I	Host address line	Either LA[28] or latched A26
HA[2]	I	Host address line	LA[29]
HA[3]	I	Host address line	LA[30]
$\overline{\text{HCS1}}$	I	Host chip select 1	$\overline{\text{LCS}n}$
$\overline{\text{HCS2}}$	I	Host chip select 2	Tie this to $V_{DD}$
HRDRW	I	Host R/W signal	LGPL1 or inverted LBCTL signal

Table 14-44. Local Bus to MSC8101 HDI16 Connections (continued)

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
$\overline{\text{HDS}}$	I	Host data strobe signal	LGPLY
HRRQ/HACK	O	Receive host request OP	As required in application
HTRQ/HREQ	O	Transmit host request OP	As required in application

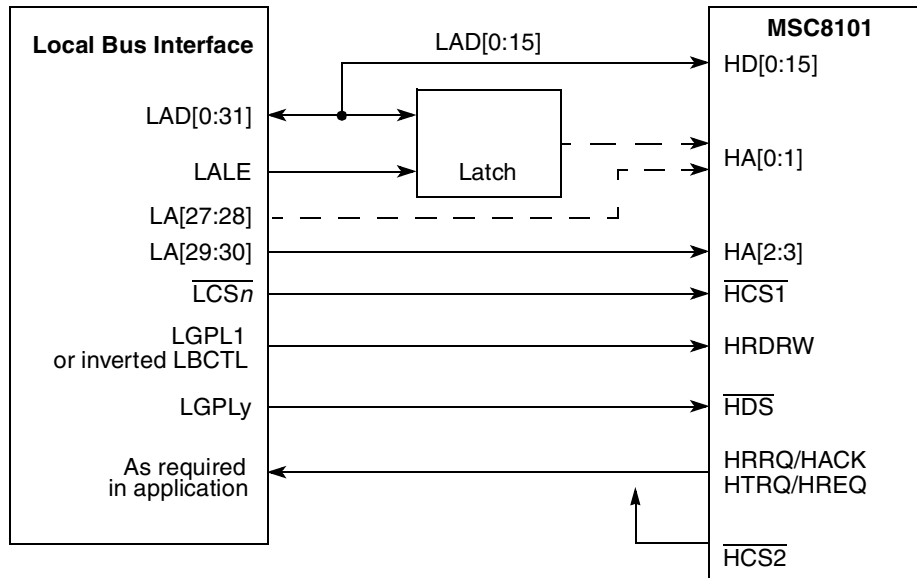


Figure 14-81. Interface to MSC8101 HDI16

The connections are specified as follows:

- One chip select,  $\overline{\text{LCSn}}$  (whatever is available in the system), is used to memory map accesses from the host local bus to the HDI16 MSC8101 HDI16 peripheral, and is connected to the HDI16 chip select line (HCS).
- This interface uses two separate general-purpose strobe lines (LGPL1 and LGPLY):
  - LGPL1 is programmed to generate the HDI16 read/write signal (HRDRW), which is typically high for a read access and low for write access. In any case, the host port requires a  $\text{HR}/\overline{\text{W}}$  signal. This can be generated by using LGPL1, and allows it to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM pattern to satisfy the AC timings at the DSP host port.
  - LGPLY is programmed to generate the HDI16 data strobe ( $\overline{\text{HDS}}$ ), which must be asserted every 16-bit read or write transaction.
- Data lines—The bus data lines ( $\overline{\text{LADn}}$ ) are directly connected to the HDI16 data lines ( $\overline{\text{HDn}}$ ).
- DMA request/service request signals (HRRQ and HTRQ)—as appropriate in the application

- Address lines—The address lines between the LBC and the HDI16 MSC8101 can either be connected in a straightforward or a specific manner to enable burst transfers across the HDI16. The connections are defined as follows and are described in more detail in the following sub-section:
  - Either LA[27] or latched value of A25 -> HA0
  - Either LA[28] or latched value of A26 -> HA1
  - LA[29] -> HA2
  - LA[30] -> HA3
- Ground lines—In order to provide the best ground plane, it is highly advised that all grounds are common and connected together.

### 14.5.6.1.3 Supporting Burst Transfers

As mentioned previously, to facilitate burst transfers the host's local bus address lines can be connected in a very specific way. First, the local bus A31 signal is not required, as the HDI16 registers are 16-bit word addressed. Secondly, local bus LA[27] and LA[28] signals can be eliminated, so that the host side transmit and receive registers wrap around the same four 16-bit word addresses.

For example, host transmit register 0 on the HDI16 peripheral (address 0x04) can be obtained by the host by accessing any of the following memory mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses as the source or destination addresses increment after each 16-bit access to the interface for all 16 transactions within that burst. By using the addressing as defined, if the first access is at 0x20, the last will be at 0x3e but, more importantly, the four host Tx/Rx registers will have been looped around four times.

### 14.5.6.1.4 Host 60x Bus: HDI16 Peripheral Interface Hardware Timings

The host UPM-controlled local bus and the HDI16 MSC8101 HDI16 host interface are both programmable. Careful programming of the host chip select registers and UPM can meet the HDI16 MSC8101 host port timings.

On any bus access the critical timing for both read and write is typically around the data latch point. For the UPM based read access, the host has the flexibility to latch data on a rising or falling LCLK edge. The falling LCLK edge is used here to latch the HDI16 data into the host MSC8101 at its earliest convenience.

After the data is latched, appropriate HDI16 port data hold time is ensured before the data strobe (DS) and chip select (CS1) are negated.

On a UPM write cycle, the critical action is in enveloping the DS assertion with CS asserted to ensure proper write data hold time after latching by the HDI16 host port.

Special attention needs to be given to both the host read and write access strobe (DS) negation times ( $\overline{\text{HDS}}$  assert).

The HDI16 MSC8101 specifies some restrictions for consecutive register access, which results in a hold off negation time for the read and write access strobes.

Rather than restrict the firmware to avoid consecutive bus accesses to host port registers, the negation hold off times should be accommodated in the UPM hardware interface settings. Additional clocks must be built into the end of UPM based cycle giving appropriate time before the next bus cycle starts.

The timings can be readily adapted to allow external decode logic to be added to support chip selects for a larger number of DSP HDI16 host ports.

### 14.5.6.2 Interfacing to MSC8102 DSI

The MSC8102 direct-slave interface (DSI) gives an external host direct access to the MSC8102. It provides the following slave interfaces to an external host:

- Asynchronous SRAM-like interface giving the host single accesses (with no external clock).
- Synchronous SSRAM-like interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8102 internal bus clock.

The DSI supports a 32- or 64-bit data bus. For connection to the local bus the DSI has to be configured in 32-bit mode. This is achieved through the DSP reset configuration.

The DSI supports two addressing modes, which are determined during the MSC8102 boot sequence. Refer to details in the MSC8102 documentation.

- Full address bus mode with HA[11:29] used in both 32-bit data mode and 64-bit data mode
- Sliding window mode with HA[14:29] used in both 32-bit data mode and 64-bit data mode

#### 14.5.6.2.1 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. As an example the dual strobe configuration is shown below.

Figure 14-82 shows the interface to the MSC8102 DSI for asynchronous mode.

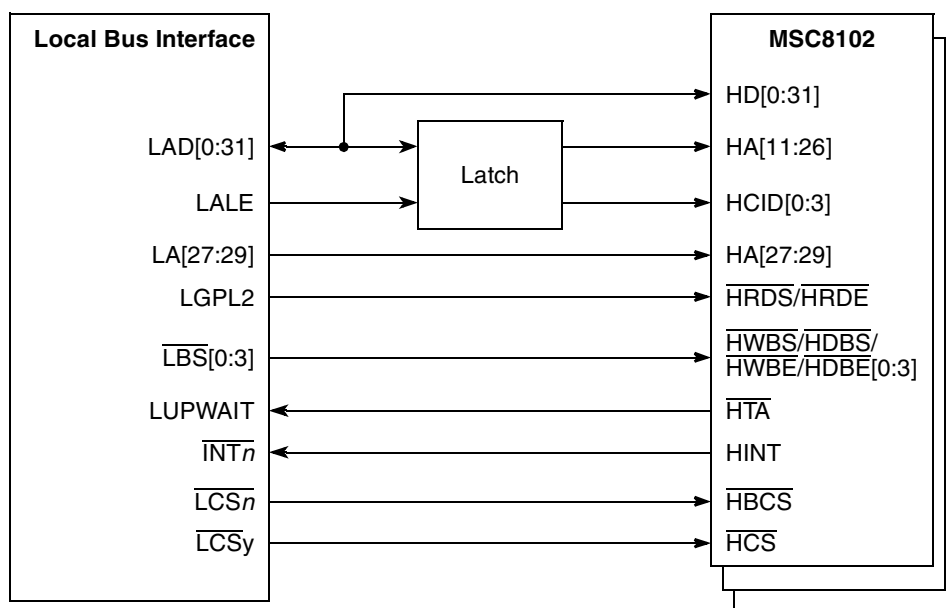


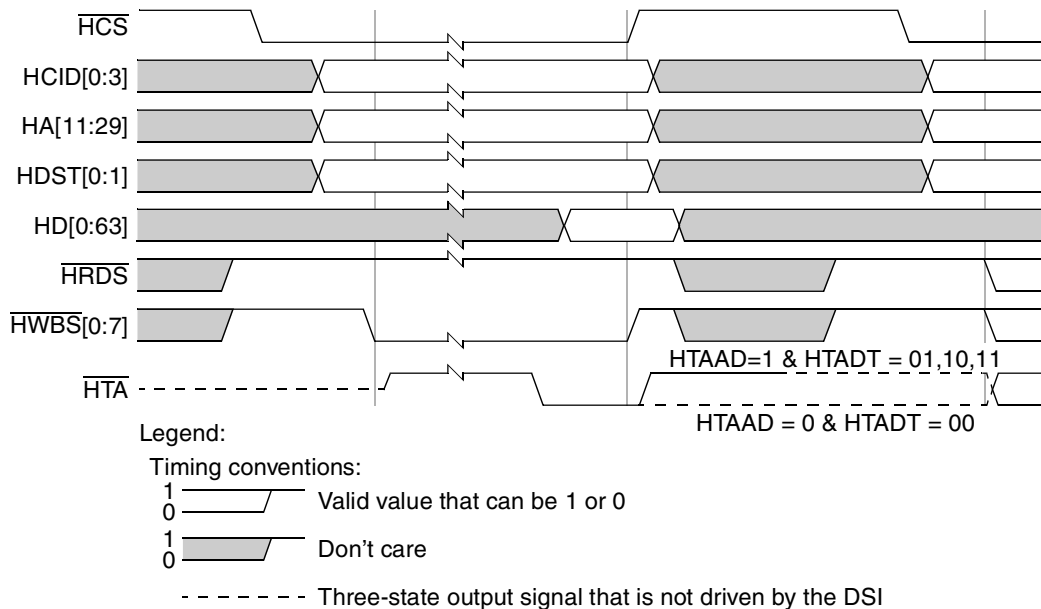
Figure 14-82. Interface to MSC8102 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used, if only relatively small amounts of data are transferred between the communications controller and the MSC8102. To allow for maximum timing flexibility, the UPM machine of the LBC should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. The UPM allows sufficient control to satisfy any requirements here.

Figure 14-83 shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0:3]) on the first falling edge of the host write byte strobe signals ( $\overline{HWBS}$ ) on which the host chip select signal ( $\overline{HCS}$ ) is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. The DSI will signal with the assertion of the host transfer acknowledge signal ( $\overline{HTA}$ ), whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating  $\overline{HWBS}$ . The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the MxMR must be cleared to interpret the correct polarity of  $\overline{HTA}$ . The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of  $\overline{HWBS}$ . In addition the assertion of  $\overline{HWBS}$ [0:3] are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[4]:HTAAD should be set to 1 and DCR[9–10]:HTADT should be defined to a value different than 00. This mode is to be used in implementations with a pull-up resistor on  $\overline{HTA}$ . The host can start its next access (back-to-back accesses) without negating  $\overline{HCS}$  between accesses. If the next access is not to the same MSC8102, then to prevent contention on the  $\overline{HTA}$  signal, the host must wait until the previous DSI stops driving  $\overline{HTA}$  before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive accesses before  $\overline{HTA}$  is actively driven to a value of 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{HTA}$  is inactive.

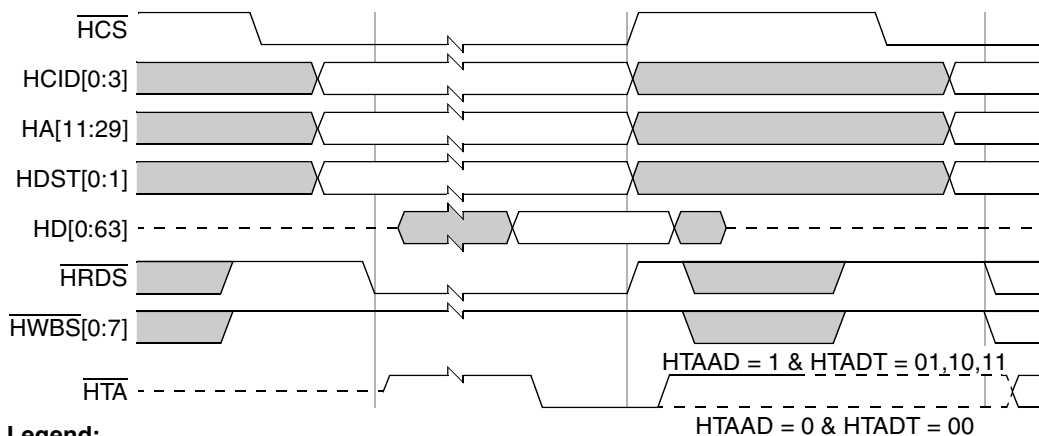


**Figure 14-83. Asynchronous Write to MSC8102 DSI**

Figure 14-84 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ( $\overline{\text{HRDS}}$ ) on which the  $\overline{\text{HCS}}$  is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. When the DCR[8]:RPE bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. The DSI signals (with the assertion of the host transfer acknowledge signal ( $\overline{\text{HTA}}$ )) that data is valid, and the host can sample the host data bus (HD) and terminate the access by negating  $\overline{\text{HRDS}}$ . If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of  $\overline{\text{HTA}}$  is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. MxMR[UWPL] has to be cleared to interpret the correct polarity of the  $\overline{\text{HTA}}$  signal.

Because the UPM is used in the mode, the DCR[4]:HTAAD should be set to 1 and the drive time control field, DCR[9–10]:HTADT, should be defined to a value different than 00. This mode is specially designed to be used for implementations with a pull-up resistor on  $\overline{\text{HTA}}$ .

The host can start its next access (back-to-back accesses) without negating the  $\overline{\text{HCS}}$  signal between accesses. If the next access is not to the same MSC8102, then to prevent contention on  $\overline{\text{HTA}}$ , the host must wait until the previous DSI stops driving  $\overline{\text{HTA}}$  before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive access before  $\overline{\text{HTA}}$  is actively driven to 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{\text{HTA}}$  is inactive.



HTAAD = 1 & HTADT = 01,10,11  
HTAAD = 0 & HTADT = 00

**Figure 14-84. Asynchronous Read from MSC8102 DSI**

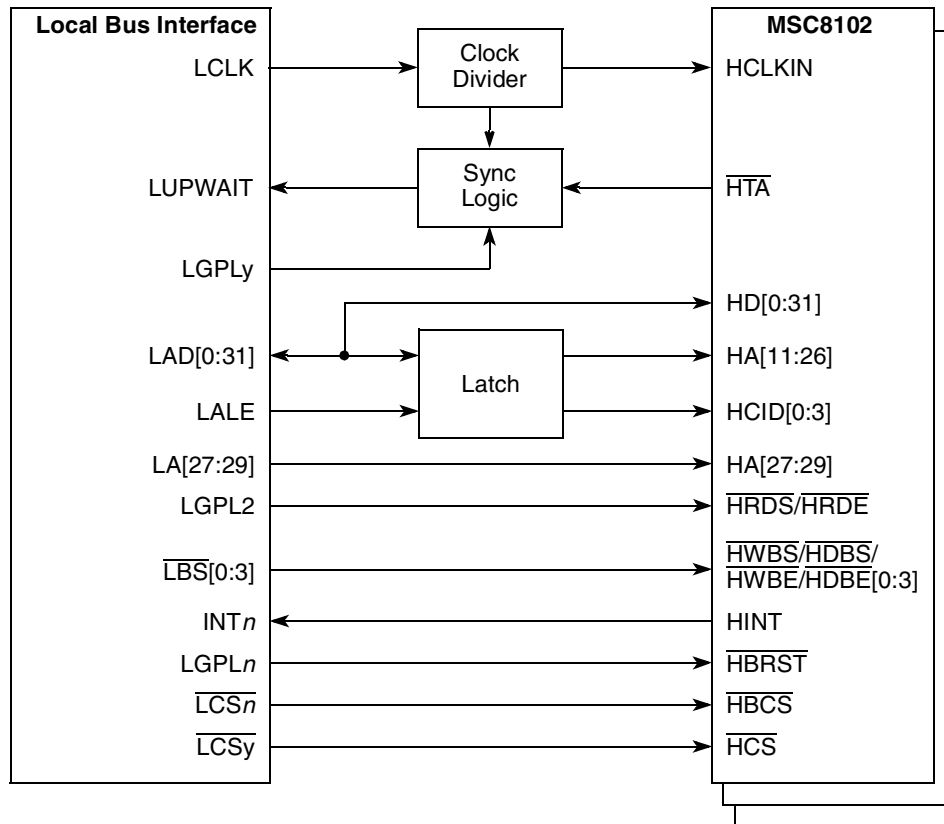
#### 14.5.6.2.2 DSI in Synchronous Mode

The synchronous SSRAM-like mode of the DSI is inherently faster than the asynchronous mode and should be used if larger amounts of data are transferred between the communications controller and the MSC8102. This will optimize the bus utilization, especially if several MSC8102s are connected to one local bus. The UPM machine of the LBC must be used to implement this interface.

Figure 14-85 shows the interface for synchronous mode. Because the DSI will assert and negate  $\overline{HTA}$  in synchronous mode even within a burst transfer on a clock-by-clock basis and because the DSI expects the host to react within one clock cycle, some tricks can be implemented to support the synchronous mode.

$\overline{HTA}$  drives LUPWAIT of the UPM.  $MxMR[UWPL]$  must be cleared to interpret the correct polarity of  $\overline{HTA}$ . Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to  $\overline{HTA}$  changes correctly within one local bus clock. Refer to Section 14.4.4.4.10, “Wait Mechanism (WAEN),” for more detailed information.

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3, or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3, or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1×, 2×, 3×, or 4×, which should be sufficient for any divider ratio that would be used in this case.



**Figure 14-85. Interface to MSC8102 DSI in Synchronous Mode**

This solution allows the local bus to react within multiple local bus clocks to the  $\overline{HTA}$  signal and be still within one DSI clock. Typically, LUPWAIT is synchronized internally, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is  $3\times$  the DSI clock, data can be sampled in the third local bus sub-clock, which is the last third of the DSI clock. If the local bus clock ratio is only  $2\times$  the DSI clock, there is a special mode, where the LUPWAIT is NOT synchronized. Refer to [Section 14.4.4.5, “Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge,”](#) for more detailed information. In this mode, data is sampled in the second sub-clock, which is the second half of the DSI clock. AC timing of LUPWAIT must be met in this mode; otherwise indeterminate behavior may occur.

The remaining issue is the synchronization of the UPM cycles to the beginning of the DSI clock cycle. Because the UPM executes  $n$  cycles for every cycle of the DSI, a mechanism must be used to ensure that the UPM changes transitions in a way that is synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal is used to control a multiplexer and to activate external synchronization logic, which uses the DSI clock to stall the UPM by asserting LUPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexer connects LUPWAIT to  $\overline{HTA}$  instead, for the rest of the bus cycle. Note that the GPL signals should be used in the inverted state of their inactive state (GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive) to start the synchronization process.



Figure 14-86 shows an example for a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary in length from one to n (clock ratio) local bus clocks.

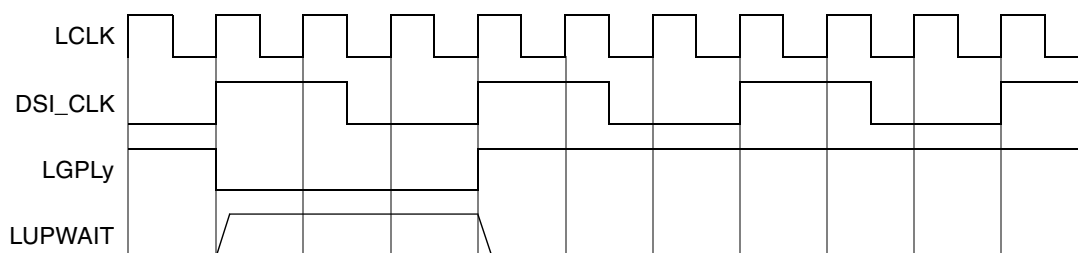


Figure 14-86. UPM Synchronization Cycle

The second column (compensation cycle) of Table 14-45 is intended to compensate for the reaction time of LUPWAIT to get in lockstep with the DSI clock. For example, if the clock divider ratio is 1:3 and the LUPWAIT reaction time is two local bus clocks, because LUPWAIT is synchronized, then one local bus clock should be inserted.

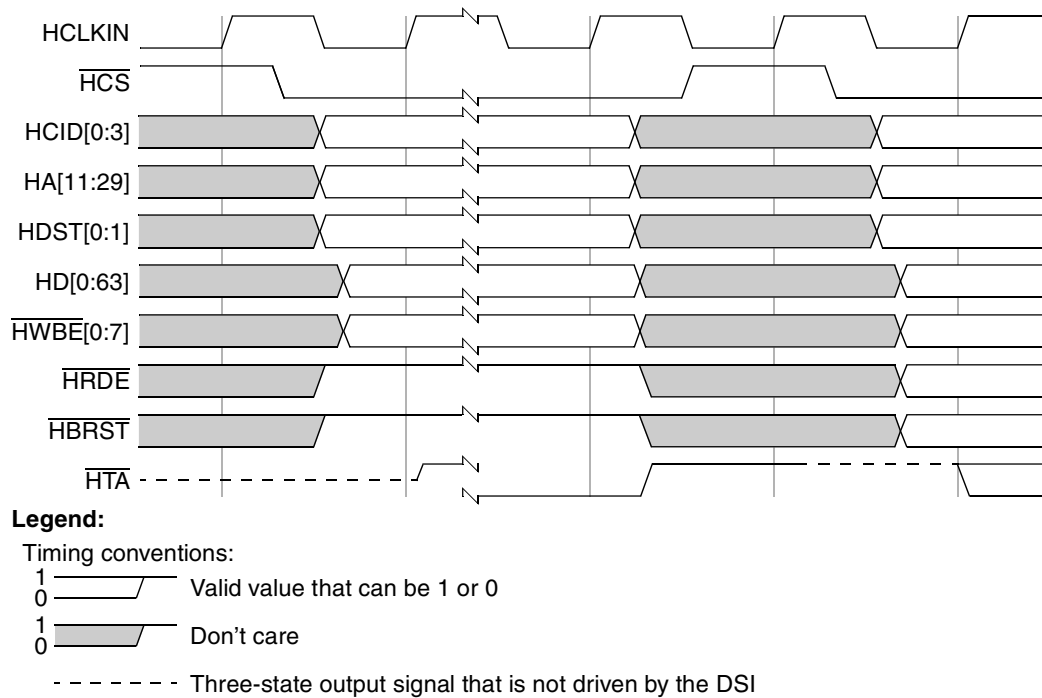
Table 14-45. UPM Synchronization Cycles

	Sync Cycle	Compensation Cycle	DSI Cycle 1	Bits
cst1–cst4				0–3
bst1–bst4				4–7
g0xx				8–11
g1tx				12–13
g2tx				14–15
g3tx				16–17
g4t1				18
g4t3	1	0		19
g5tx				20–21
redo[0]	0	0	1	22
redo[1]	0	0	0	23
loop	0	0		24
exen	0			25
amx0	0	0		26
amx1	0	0		27
na	0			28
uta	0			29
todt	0			30
last	0			31

This section describes synchronous single write and read, and synchronous burst write and read operations. The local bus supports the DSI single strobe as well as the DSI double strobes of operation. The dual strobe configuration is shown as an example.

## Synchronous Single Write

Figure 14-87 shows a synchronous single write access.

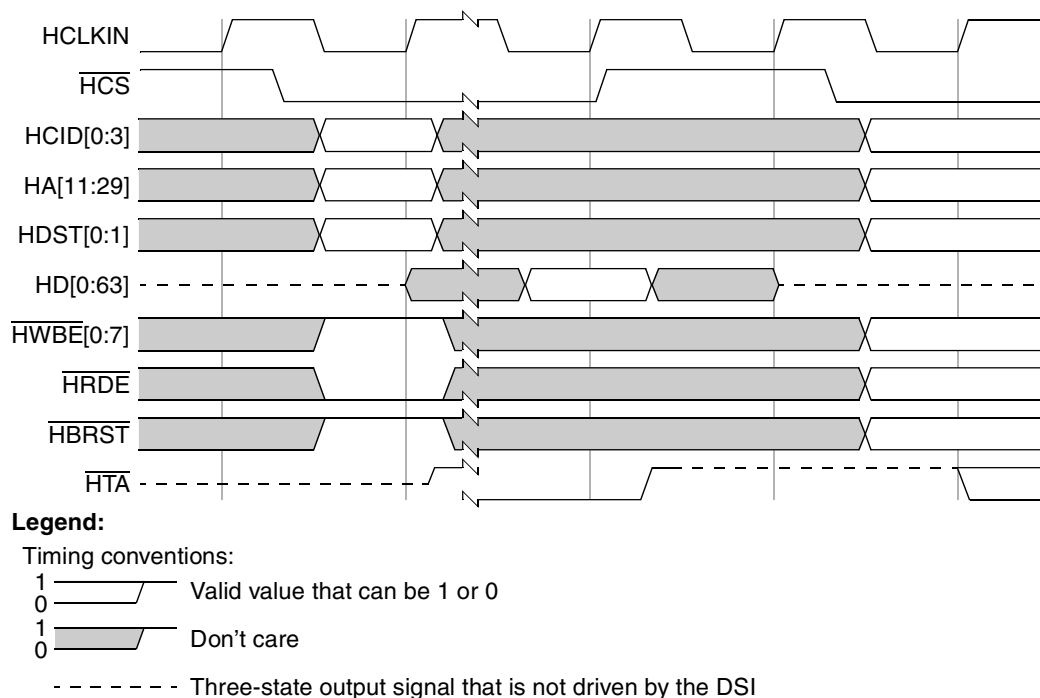


**Figure 14-87. Synchronous Single Write to MSC8102 DSI**

The DSI samples  $\overline{HA}$ ,  $\overline{HDST}$ ,  $\overline{HCID}$ ,  $\overline{HD}$ ,  $\overline{HWBE}$ ,  $\overline{HRDE}$ , and  $\overline{HBRST}$  on the first HCLKIN rising edge on which  $\overline{HCS}$  is asserted. If  $\overline{HCID}[0:3]$  match the CHIPID value, the DSI is accessed. At least one  $\overline{HWBE}$  signal is asserted, and  $\overline{HRDE}$  and  $\overline{HBRST}$  are negated. Assertion of  $\overline{HTA}$  indicates that the DSI is ready to complete the current access and the host must terminate this access. Because  $\overline{HTA}$  is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until  $\overline{HTA}$  goes to 0 and then the UPM continues in its pattern. Typically,  $\overline{HTA}$  is asserted immediately. If the write buffer is full,  $\overline{HTA}$  assertion is delayed.  $\overline{HTA}$  is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately on the next HCLKIN rising edge without negating  $\overline{HCS}$  between accesses. If the next access is not to the same MSC8102, then, to prevent contention on  $\overline{HTA}$ , the host must wait to access the next device until the previous DSI stops driving  $\overline{HTA}$ . The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{HTA}$  is inactive.

## Synchronous Single Read

Figure 14-88 shows a synchronous single read access.

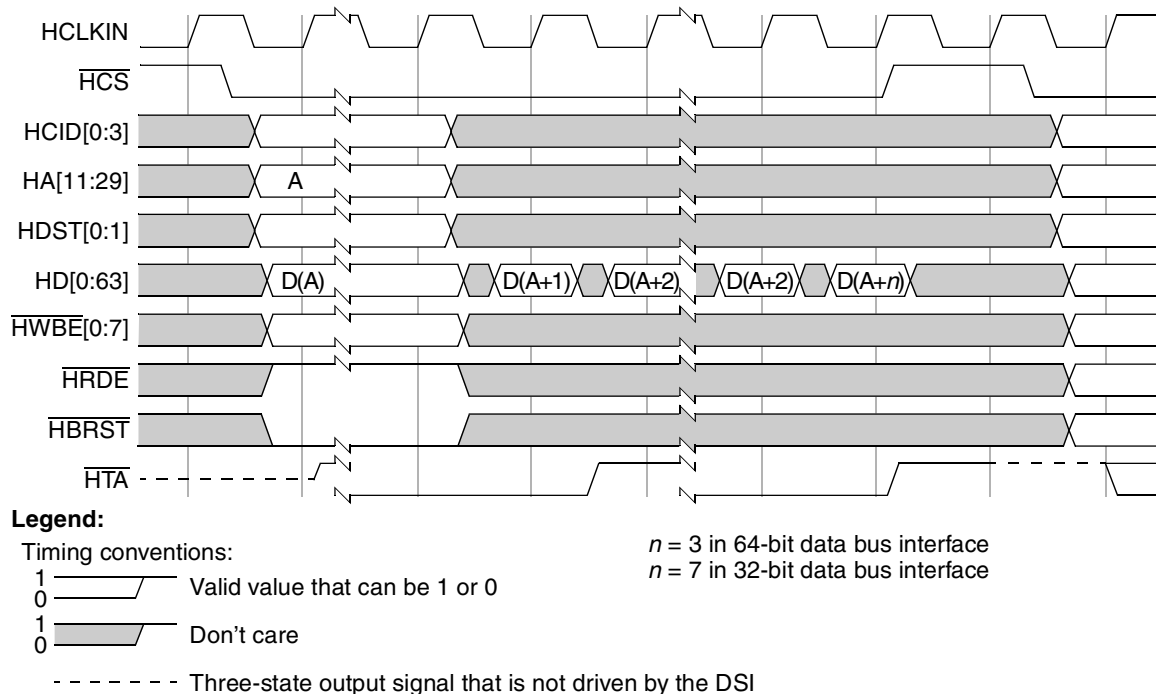


**Figure 14-88. Synchronous Single Read from MSC8102 DSI**

The DSI samples HA, HDST, HCID,  $\overline{HWBE}$ ,  $\overline{HRDE}$ , and  $\overline{HBRST}$  on the first HCLKIN rising edge on which  $\overline{HCS}$  is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed.  $\overline{HRDE}$  is asserted, and  $\overline{HWBE}$  and  $\overline{HBRST}$  are negated. If DCR[8]:RPE is set (see MSC8102 documentation), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of  $\overline{HTA}$  indicates that data is valid and the host must sample the HD and terminate the access. Because  $\overline{HTA}$  is connected to the UPM LUPWAIT signal, all local bus signals are frozen until  $\overline{HTA}$  goes to 0; then the UPM continues in its pattern.  $\overline{HTA}$  is asserted earlier when the data for this access is already prefetched to the read buffer. It is asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating  $\overline{HCS}$  between accesses. If the next access is not to the same MSC8102, then, to prevent contention on  $\overline{HTA}$ , the host must wait to access the next device until the previous DSI stops driving  $\overline{HTA}$ . The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{HTA}$  is inactive.

## Synchronous Burst Write

Figure 14-89 shows a synchronous burst write access.

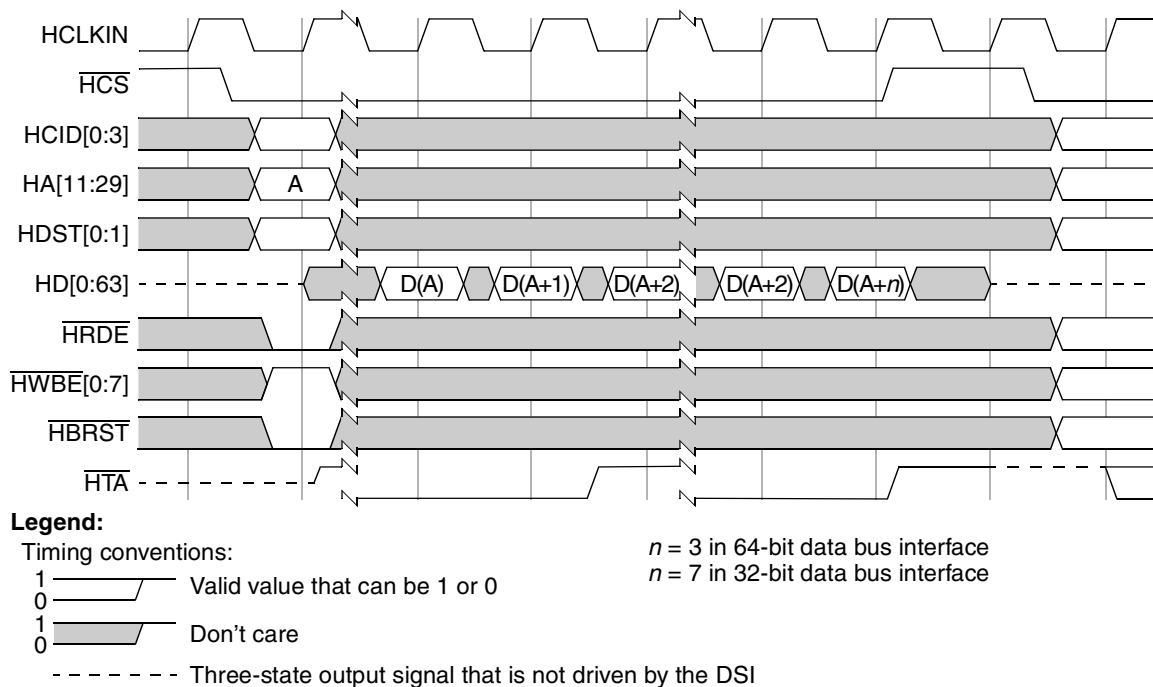


**Figure 14-89. Synchronous Burst Write to MSC8102 DSI**

The DSI samples HA, HDST, HCID, HD,  $\overline{\text{HWBE}}$ ,  $\overline{\text{HRDE}}$ , and  $\overline{\text{HBRST}}$  on the first HCLKIN rising edge on which  $\overline{\text{HCS}}$  is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.  $\overline{\text{HWBE}}$  are asserted,  $\overline{\text{HBRST}}$  is asserted, and  $\overline{\text{HRDE}}$  is negated. Assertion of  $\overline{\text{HTA}}$  indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically  $\overline{\text{HTA}}$  is asserted immediately for each beat of the access. If the write buffer is full,  $\overline{\text{HTA}}$  assertion is delayed. Because  $\overline{\text{HTA}}$  is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until  $\overline{\text{HTA}}$  goes to 0 and then the UPM continues in its pattern. After the last beat of the access,  $\overline{\text{HTA}}$  is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating  $\overline{\text{HCS}}$  between accesses. If the next access is not to the same MSC8102, then, to prevent contention on  $\overline{\text{HTA}}$ , the host must wait to access the next device until the previous DSI stops driving  $\overline{\text{HTA}}$ . The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{\text{HTA}}$  is inactive.

## Synchronous Burst Read

Figure 14-90 shows a synchronous burst read access. The DSI samples HA, HDST, HCID,  $\overline{\text{HWBE}}$ ,  $\overline{\text{HRDE}}$ , and  $\overline{\text{HBRST}}$  on the first HCLKIN rising edge on which  $\overline{\text{HCS}}$  is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.



**Figure 14-90. Synchronous Burst Read from MSC8102 DSI**

$\overline{\text{HRDE}}$  and  $\overline{\text{HBRST}}$  are asserted and  $\overline{\text{HWBE}}$  are negated. When the DCR[8]:RPE bit (see the MSC8102 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of  $\overline{\text{HTA}}$  indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because  $\overline{\text{HTA}}$  is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until  $\overline{\text{HTA}}$  goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The  $\overline{\text{HTA}}$  is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access,  $\overline{\text{HTA}}$  remains asserted until the end of the access. After the last beat of the access,  $\overline{\text{HTA}}$  is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating  $\overline{\text{HCS}}$  between accesses. If the next access is not to the same MSC8102, to prevent contention on  $\overline{\text{HTA}}$ , the host must wait to access the next device until the previous DSI stops driving the  $\overline{\text{HTA}}$  signal. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee that  $\overline{\text{HTA}}$  is inactive.

### 14.5.6.2.3 Broadcast Accesses

Using  $\overline{\text{HBCS}}$ , a host can share one chip-select signal between multiple MSC8102 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its  $\overline{\text{HTA}}$  signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode HCID[0:3].

Note that broadcasting is allowed only for write accesses.

The DSI sets the DSI error register (DER) OVF bit if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it.

#### NOTE

To avoid overflow when accessing DSI registers during broadcast accesses, wait at least 10 host clock cycles in synchronous mode or 8 internal clock cycles in asynchronous mode between each DSI register access.

To avoid data corruption, if DER[0]:OVF is set, no broadcast access is written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, DER[0]:OVF must first be read and reset if it is set.

#### NOTE

In asynchronous mode, write data from a previous access (even a normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers, depending on the type of previous access. This can be achieved by performing a read access prior to the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode,  $\overline{HWBS}[0:3]/\overline{HDBS}[0:3]$  assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8102 Technical Data* sheet.
- In synchronous mode single access, the host must wait 1 cycle before terminating the access. Access signals must be in the same valid state during two positive edges of the host clock cycles. Access duration is two clock cycles (the DSI may translate accesses lasting longer than two clock cycles as two or more back-to-back accesses).
- In synchronous mode burst accesses, broadcast accesses are not allowed.

# Chapter 15

## Enhanced Three-Speed Ethernet Controllers

### 15.1 Overview

#### NOTE

Although this chapter describes the implementation of and interaction between four eTSEC interfaces, the MPC8533E only offers two (eTSEC interfaces 1 and 3). All references to eTSECs 2 and 4 should be disregarded for the MPC8533E.

The enhanced three-speed Ethernet controllers (eTSECs) of the device interface to 10 Mbps, 100 Mbps, and 1 Gbps Ethernet/IEEE 802.3 networks and devices featuring generic 8-bit FIFO ports. For Ethernet, an external PHY or SerDes device is required to complete the interface to the media. Each eTSEC supports multiple standard media-independent interfaces, of which the FIFO interface bypasses the Ethernet MAC. Two eTSECs are available, providing flexible options for connectivity and control access at different speeds.

The eTSEC provides the flexibility to accelerate the identification and retrieval of standard and non-standard protocols carried over Ethernet, including both IP versions 4 and 6 and TCP/UDP. CPU-intensive parsing and checksum operations can be optionally off-loaded to an eTSEC to accelerate existing TCP/IP stacks. On transmission, varying fractions of link bandwidth can be allocated to each of multiple transmit queues through a modified weighted round-robin scheduler. On receive, an arbitrary set of queue selection rules can be programmed into each eTSEC to implement flexible quality of service or firewall strategies based on high-level protocol identification. Without enabling these advanced features, each eTSEC emulates a PowerQUICC III TSEC, allowing existing driver software to be re-used with minimal change. Each eTSEC is organized as shown in [Figure 15-1](#).

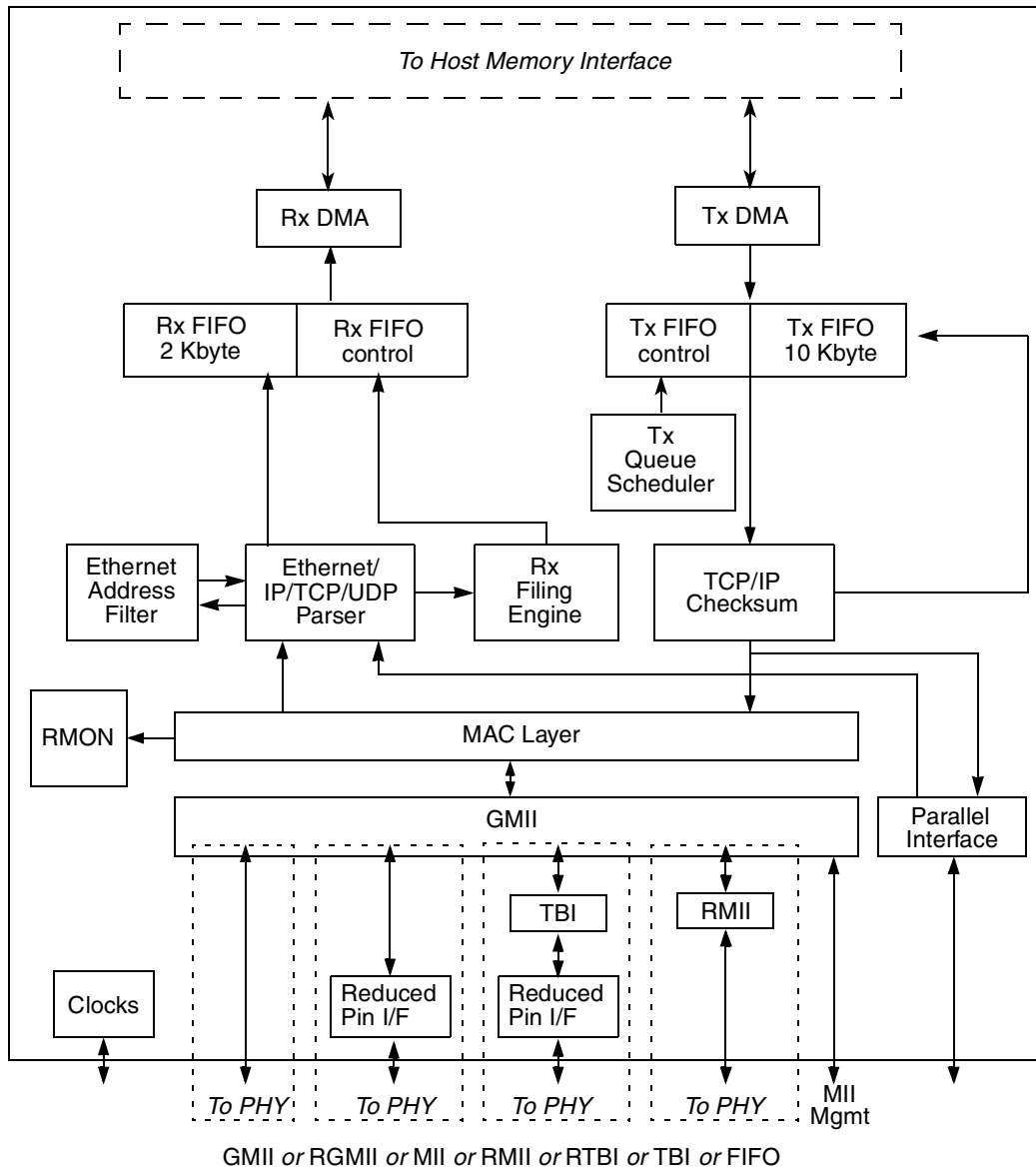


Figure 15-1. eTSEC Block Diagram

## 15.2 Features

The eTSECs of the device include these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant
- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE 802.3 GMII
  - 1000 Mbps full-duplex IEEE 802.3 GMII
  - 10/100 Mbps IEEE 802.3 MII and RMII
  - 10/100 Mbps RGMII



- 1000 Mbps full-duplex RGMII and RTBI
- 1000 Mbps IEEE 802.3z TBI
- Single-clock TBI
- Support for four full-duplex FIFO interface modes
  - Two 8-bit modes—GMII style and encoded packet
  - Inter-packet and intra-packet flow control
  - Optional CRC-32 generation and checking
  - Minimal glue logic required to support POS PHY Level 3 conversion
  - TCP/IP off-load and QoS features available in all FIFO modes
- TCP/IP off-load
  - IP v4 and IP v6 header recognition on receive
  - IP v4 header checksum verification and generation
  - TCP and UDP checksum verification and generation
  - Per-packet configurable off-load
  - Recognition of VLAN, stacked-VLAN, 802.2, PPPoE session, MPLS stacks, and ESP/AH IP-Security headers
- Quality of service (QoS) support
  - Transmission from up to eight queues
    - Priority-based queue selection
    - Modified weighted round-robin queue selection with fair bandwidth allocation
  - Reception to up to eight physical queues
    - 64 virtual receive queues overlaid on 8 physical buffer descriptor rings
    - Table-oriented queue filing strategy based on 16 header fields or flags
    - Frame rejection support for filtering applications
    - Filing based on Ethernet, IP, and TCP/UDP properties, including VLAN fields, Ether-type, IP protocol type, IP TOS or differentiated services, IP source and destination addresses, TCP/UDP port numbers
- Interrupt coalescing
  - Packet-count-based thresholds for both receive and transmit
  - Timer-based thresholds
- Full- and half-duplex Ethernet support (1000 Mbps supports only full duplex):
  - IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
  - Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) tags and priority
  - VLAN insertion and deletion
    - Per-frame VLAN control word or default VLAN for each eTSEC
    - Extracted VLAN control word passed to software separately

- Programmable VLAN tag to support metropolitan bridging
- Retransmission following a collision
- Support for CRC generation and verification of inbound/outbound packets
- Programmable Ethernet preamble insertion and extraction of up to 7 bytes
- MAC address recognition:
  - Exact match on primary and virtual 48-bit unicast addresses
    - VRRP and HSRP support for seamless router fail-over
    - In addition to primary station address, up to fifteen additional exact-match MAC addresses supported
  - Broadcast address (accept/reject)
  - Hash table match on up to 256 unicast/multicast or 512 multicast-only addresses
  - Promiscuous mode
- Remote network monitoring (RMON) statistics support
  - 32-bit byte counters
  - Carry/Overflow of counter interrupts
- Backward compatibility with MPC8540E/MPC8560E (PowerQUICC III) TSEC
  - PowerQUICC III buffer descriptor (BD) format and rings supported
  - Common register memory map, with specific exceptions:
    - Out-of-sequence transmit BD not supported
    - Internal DMA BD pointers and data counts not visible
    - MINFLR register not supported
  - Reset state of eTSEC defaults to common PowerQUICC III TSEC subset
  - TSEC\_ID register permits TSEC versus enhanced TSEC differentiation

### 15.3 Modes of Operation

The eTSEC's primary operational modes are the following:

- Ethernet and FIFO operation
 

The ECNTRL register's FIFO mode enable bit (ECNTRL[FIFM]) allows bypass of the Ethernet MAC and enables I/O through the FIFO interface sharing the normal GMII signals. Each eTSEC supports an 8-bit FIFO interface independently. If configured in FIFO mode, the FIFOCFG register determines operation. In FIFO mode data is transferred synchronously with respect to the external data clock, whose maximum is defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and a ratio of 3.2:1 (platform:TxClk) in encoded mode.
- Full- and half-duplex operation
 

This is determined by the MACCFG2 register's full-duplex bit (MACCFG2[Full Duplex]). Full-duplex mode is intended for use on point-to-point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

If configured in half-duplex mode (10- and 100-Mbps operation; MACCFG2[Full Duplex] is cleared), the MAC complies with the IEEE CSMA/CD access method.

If configured in full-duplex mode (10/100/1000 Mbps operation; MACCFG2[Full Duplex] is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.

- 10- and 100-Mbps MII interface operation

The MAC-PHY interface operates in MII mode by setting MACCFG2[I/F Mode] = 01. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the TSEC<sub>n</sub>\_TX\_CLK and TSEC<sub>n</sub>\_RX\_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed, or it may be controlled by software using the serial management interface (MDC/MDIO signals) to the transceiver.

Clause 22.2.4 of the IEEE 802.3 specification describes the MII management interface.

- 10- and 100-Mbps RMII interface operation

The RMII is the reduced media-independent interface defined by the RMII Consortium (March 1998) for 10/100 Mbps operation. The speed of operation is determined by the TSEC<sub>n</sub>\_TX\_CLK signal, which is driven by the transceiver.

- 1000 Mbps GMII and TBI interface operation

The MAC-PHY interface operates in GMII mode by setting MACCFG2[I/F Mode] = 10. The GMII is the gigabit media-independent interface defined by the 802.3 standard for 1000-Mbps operation.

Independently, the MAC-PHY interface can also operate in TBI mode. Note that either the TBI or GMII interface is chosen, not both at the same time. TBI is the 10-bit interface which contains PCS functions (10-bit encoding/decoding) as defined by the 802.3 standard.

In reduced-pin count mode (RGMII or RTBI), the MAC remains configured in GMII or TBI but the eTSEC muxes and decodes the input signals and provides the MAC with the expected interface. eTSEC provides the TSEC<sub>n</sub>\_GTX\_CLK to the PHY in either GMII or TBI mode of operation.

- MAC address recognition options

The options supported are promiscuous, broadcast, exact unicast address match, exact unicast virtual address match to support router redundancy, and multicast hash match. For detailed descriptions refer to [Section 15.6.3.7, “Frame Recognition.”](#)

eTSEC supports automatic LAN-initiated wake-up during power management via the AMD Magic Packet™ protocol, as described in [Section 15.6.3.8, “Magic Packet Mode.”](#)

- Receive frame parsing options

Frame parsing options are to disable parsing (no TCP/IP off-load), IP header parsing, and TCP or UDP parsing. Parsing must be enabled to make use of receive queue filing algorithms. The options are detailed in [Section 15.6.4, “TCP/IP Off-Load.”](#)

- Receive queue selection options

Received frames are by default sent to a single buffer descriptor ring. If multiple receive queues are enabled, a receive queue filer can be programmed with selection criteria to differentiate

received frames and file them to different buffer descriptor rings. See [Section 15.6.5, “Quality of Service \(QoS\) Provision,”](#) for detailed descriptions.

- TCP/IP transmit options

Frames for transmission may be sent as-is, with IP header processing, or TCP header processing. The transmit buffer descriptors, described in [Section 15.6.7.2, “Transmit Data Buffer Descriptors \(TxBD\),”](#) enable these options and operate with parameters prepended to frame buffers, as described in [Section 15.6.4, “TCP/IP Off-Load.”](#)

- Transmit queue selection options

The options supported are single transmit queue, priority-based queue selection, and modified weighted round-robin queueing. These options are described further in [Section 15.5.3.2.1, “Transmit Control Register \(TCTRL\).”](#)

- RMON support

Standard Ethernet interface management information base (MIBs) can be generated via the RMON MIB counters.

- Internal loop back supported for all interfaces except when configured for half-duplex operation

Internal loop back mode is selected via the loop back bit in the MACCFG1 register. See [Section 15.7.1, “Interface Mode Configuration,”](#) for details.

## 15.4 External Signals Description

This section defines the eTSEC interface signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention. (That is, TxD[7:0] means 0 is the lsb.) Note that except for external physical interfaces the buses and registers follow a big-endian format, where 0 denotes the msb.

Each eTSEC network interface supports multiple options:

- The MII option requires 18 I/O signals (including the MDIO and MDC MII management interface) and supports both a data and a management interface to the PHY (transceiver) device. The MII option supports both 10- and 100-Mbps Ethernet rates.
- The GMII option is a superset of the MII signals and supports a 1000-Mbps Ethernet rate.
- The TBI interface shares signals with the GMII interface signals.
- The RGMII, RTBI, and RMII options are reduced-pin implementations of the GMII, TBI, and MII interfaces, respectively.
- Finally, the FIFO interfaces share the GMII signals—8 bits of data plus 3 bits of control signals.

Table 15-1 lists the network interface signals.

**Table 15-1. eTSEC<sub>n</sub> Network Interface Signal Properties**

Signal Name	Function	Reset State
TSEC <sub>n</sub> _COL	MII—collision, input FIFO—transmit flow control, input	—
TSEC <sub>n</sub> _CRS	MII—carrier sense, input TBI—signal detect, input FIFO—receive flow control, output	—
TSEC <sub>n</sub> _GTX_CLK	RTBI, RGMII—invetered transmit clock feedback, output TBI FIFO—continuous transmit clock feedback, output GMII, MII, RMII—transmit clock feedback when transmission is enabled, zero otherwise, output	0
EC_GTX_CLK <sup>125</sup>	Oscillator source for GMII, TBI, RGMII, RTBI transmit clock, input, shared by eTSEC1 and eTSEC3	—
EC_MDC	TBI, RTBI, GMII, RGMII, MII, RMII—Management clock, output.	0
EC_MDIO	TBI, RTBI, GMII, RGMII, MII, RMII—Management data, bidirectional.	Hi-Z (input)
TSEC <sub>n</sub> _RX_CLK	GMII, MII, RGMII—receive clock, input TBI—PMA receive clock 0, input FIFO—receive clock, input	—
TSEC <sub>n</sub> _RX_DV	GMII, MII—receive data valid, input TBI—receive code group (RCG) bit 8, input RGMII (RX_CLK rising)—receive data valid, input RGMII (RX_CLK falling)—receive error, input RTBI (RX_CLK rising)—receive code group (RCG) bit 4, input RTBI (RX_CLK falling)—receive code group (RCG) bit 9, input RMII—CRS_DV carrier sense/data valid, input FIFO—receive data valid or receive control bit, input	—
TSEC <sub>n</sub> _RXD[7:4]	GMII—receive data bits 7:4 input TBI—RCG bits 7:4, input FIFO—receive data bits 7:4 input MII, RGMII, RTBI, RMII—unused	—
TSEC <sub>n</sub> _RXD[3:0]	GMII, MII—Receive data bits 3:0, input TBI—RGC bits 3:0, input RGMII (RX_CLK rising)—Receive data bits 3:0, input RGMII (RX_CLK falling)—Receive data bits 7:4, input RTBI (RX_CLK rising)—RCG bits 3:0, input RTBI (RX_CLK falling)—RCG bits 8:5, input RMII—RXD[1:0] receive data bits, input RMII—RXD[3:2] are unused FIFO—Receive data bits 3:0, input	—
TSEC <sub>n</sub> _RX_ER	GMII, MII, RMII—Receive error, input TBI—RGC bit 9, input FIFO—Receive error or receive frame control bit, input RGMII, RTBI—Unused, output driven zero	—

Table 15-1. eTSEC<sub>n</sub> Network Interface Signal Properties (continued)

Signal Name	Function	Reset State
TSEC <sub>n</sub> _TX_CLK	MII—transmit clock, input TBI—PMA receive clock 1, input RMII—reference transmit and receive clock, input FIFO—transmit clock, input RGMII, RTBI—unused	—
TSEC <sub>n</sub> _TXD[7:4]	GMII—transmit data bit 7:4, output TBI—transmit code group (TCG) bit 7:4, output FIFO—transmit data bit 7:4, output MII, RGMII, RTBI, RMII—unused, output driven zero	0000
TSEC <sub>n</sub> _TXD[3:0]	GMII, MII—Transmit data bits 3:0, output TBI—TCG bits 3:0, output RGMII (TX_CLK rising)—Transmit data bits 3:0, output RGMII (TX_CLK falling)—Transmit data bits 7:4, output RTBI (TX_CLK rising)—TCG bits 3:0, output RTBI (TX_CLK falling)—TCG bits 8:5, output RMII—TXD[1:0] transmit data bits, output RMII—TXD[3:2] unused, output driven zero FIFO—Transmit data bits 3:0, output	0000
TSEC <sub>n</sub> _TX_ER	GMII, MII—transmit error, output RGMII, RTBI, RMII—unused, output driven zero TBI—TCG bit 9, output FIFO—transmit error or transmit frame control bit, output	0
TSEC <sub>n</sub> _TX_EN	GMII, MII, RMII—Transmit data valid, output TBI—TCG bit 8, output RGMII (TX_CLK rising)—Transmit data enabled, output RGMII (TX_CLK falling)—Transmit error, output RTBI (TX_CLK rising)—TCG bit 4, output RTBI (TX_CLK falling)—TCG bit 9, output FIFO—Transmit data valid or transmit control bit, output	0

### 15.4.1 Detailed Signal Descriptions

Below is a description of the eTSEC interface signals. For RGMII mode details please refer to the Hewlett-Packard reduced gigabit media-independent interface (RGMII) specification version 1.2a, dated 9/22/2000. RMII mode details follow the RMII Consortium Specification, dated 3/20/1998. All other modes follow the IEEE 802.3 standard, 2000 Edition. Input signals not used are internally disabled. Except for TSEC<sub>n</sub>\_GTX\_CLK, output signals not used are driven low.

Table 15-2. eTSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description
TSEC <sub>n</sub> _COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.
		<p><b>State Meaning</b> Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists. In FIFO mode this signal is used to effect flow control on the transmitter.</p> <p>This signal is not used in the following modes:</p> <ul style="list-style-type: none"> <li>• RMII</li> <li>• GMII</li> <li>• TBI</li> <li>• RTBI</li> <li>• RGMII</li> </ul>
		<p><b>Timing</b> Asserted/Negated—This signal is not required to transition synchronously with TSEC<sub>n</sub>_TX_CLK or TSEC<sub>n</sub>_RX_CLK.</p>
TSEC <sub>n</sub> _CRS	I	Carrier sense input. In TBI and RTBI modes, this signal is used as SDET (signal detect). In TBI mode SDET must be tied high externally on the board. In RTBI mode SDET is tied high internally. This signal is not used in the following modes:
		<ul style="list-style-type: none"> <li>• RMII</li> <li>• GMII</li> <li>• RGMII</li> </ul>
		<p><b>State Meaning</b> Asserted/Negated—In MII mode, TSEC<sub>n</sub>_TX_CLK is asserted while the transmit or receive medium is not idle. In the event of a collision, TSEC<sub>n</sub>_CRS must remain asserted for the duration of the collision.</p>
	<p><b>Timing</b> Asserted/Negated—This signal is not required to transition synchronously with TSEC<sub>n</sub>_TX_CLK or TSEC<sub>n</sub>_RX_CLK.</p>	
	O	Receiver flow control signal in FIFO mode. This signal is not used in the eTSEC Ethernet modes.
		<p><b>State Meaning</b> Asserted/Negated—TSEC<sub>n</sub>_CRS is asserted while the FIFO receiver is unprepared to accept additional receive data.</p>
<p><b>Timing</b> Asserted/Negated—This signal transitions synchronously with TSEC<sub>n</sub>_RX_CLK.</p>		
TSEC <sub>n</sub> _GTX_CLK	O	<p>Gigabit transmit clock. This signal is an output from the eTSEC into the PHY. TSEC<sub>n</sub>_GTX_CLK is a 125-MHz clock that provides a timing reference for TX_EN, TXD, and TX_ER in the following modes:</p> <ul style="list-style-type: none"> <li>• GMII</li> <li>• TBI</li> <li>• RTBI</li> </ul> <p>In RGMII mode, TSEC<sub>n</sub>_GTX_CLK becomes the transmit clock and provides timing reference during 1000Base-T (125 MHz), 100Base-T (25 MHz) and 10Base-T (2.5 MHz) transmissions. This signal feeds back the uninverted transmit clock in MII or FIFO modes, but feeds back an inverted transmit clock in RTBI or RGMII modes.</p> <p>This signal is driven low unless transmission is enabled, or the eTSEC is in TBI or FIFO mode.</p>

Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
EC_GTX_CLK125	I	Gigabit transmit 125-MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY. EC_GTX_CLK125 is a 125-MHz input into the eTSEC and is used to generate all 125-MHz related signals and clocks in the following modes: <ul style="list-style-type: none"> <li>• GMII</li> <li>• TBI</li> <li>• RTBI</li> <li>• RGMII</li> </ul> This input is not used in MII, or RMII or FIFO modes.	
EC_MDC	O	Management data clock. This signal is a clock (typically 2.5 MHz) supplied by the MAC in the following modes: <ul style="list-style-type: none"> <li>• MII</li> <li>• RMII</li> <li>• TBI</li> <li>• RTBI</li> <li>• GMII</li> <li>• RGMII</li> </ul> (IEEE set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed.) The frequency can be modified by writing to MIIMCFG[28:31] of the eTSEC1 controller.	
EC_MDIO	I/O	Management data input/output, BIDI	
		<b>State Meaning</b>	Asserted/Negated—In MII, RMII, TBI, RTBI, GMII, or RGMII mode EC_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles. Addressed using eTSEC1 memory-mapped registers.
		<b>Timing</b>	Asserted/Negated—This signal is required to be synchronous with the EC_MDC signal.
TSEC <sub>n</sub> _RX_CLK	I	Receive clock. In GMII, MII, or RGMII mode, the receive clock TSEC <sub>n</sub> _RX_CLK is a continuous clock (2.5, 25, or 125 MHz) that provides a timing reference for TSEC <sub>n</sub> _RX_DV, TSEC <sub>n</sub> _RXD, and TSEC <sub>n</sub> _RX_ER. In TBI mode, TSEC <sub>n</sub> _RX_CLK is the input for a 62.5 MHz PMA receive clock, 0 split phase with PMA_RX_CLK1 and is supplied by the SerDes. In RTBI mode it is a 125-MHz receive clock. In RMII mode this clock is not used for the receive clock, as RMII uses a shared reference clock. In FIFO mode the receive clock is a continuous clock whose maximum is defined by a ratio of 4.2:1 (platform:RxClk) in GMII mode and a ratio of 3.2:1 (platform:RxClk) in encoded mode.	
TSEC <sub>n</sub> _RX_DV	I	Receive data valid. In GMII or MII mode, if TSEC <sub>n</sub> _RX_DV is asserted, the PHY is indicating that valid data is present on the GMII and MII interfaces. In RGMII mode, TSEC <sub>n</sub> _RX_DV becomes RX_CTL. The RX_DV and RX_ERR are received on this signal on the rising and falling edges of TSEC <sub>n</sub> _RX_CLK. In TBI mode, TSEC <sub>n</sub> _RX_DV represents receive code group (RCG) bit 8. Together, with RCG[9] and RCG[7:0], they represents the 10-bit encoded symbol of GMII receive signals. In RTBI mode, TSEC <sub>n</sub> _RX_DV represents receive code group (RCG) bit 4 and 9. On the positive edge of the TSEC <sub>n</sub> _RX_CLK, RCG[4] and RCG[3:0] represent the first half of the 10-bit encoded symbol. On the negative edge of the TSEC <sub>n</sub> _RX_CLK, RCG[9] and RCG[8:5] represent the second half of the 10-bit encoded symbol. In RMII mode the PHY asserts TSEC <sub>n</sub> _RX_DV (CRS_DV) when the receive medium is non-idle. This signal asserts asynchronously with respect to the RMII reference clock, but negates synchronously to indicate loss of carrier. In FIFO mode TSEC <sub>n</sub> _RX_DV is used to indicate valid data (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols).	



Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description				
TSEC <sub>n</sub> _RXD[7:0]	I	<p>Receive data in. In GMII mode, TSEC<sub>n</sub>_RXD[7:4] with TSEC<sub>n</sub>_RXD[3:0], represent one complete octet of data to be transferred from the PHY to the MAC when TSEC<sub>n</sub>_RX_DV is asserted. In TBI mode, TSEC<sub>n</sub>_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals.</p> <p>In GMII or MII mode, TSEC<sub>n</sub>_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSEC<sub>n</sub>_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While TSEC<sub>n</sub>_RX_DV is not asserted, TSEC<sub>n</sub>_RXD has no meaning.</p> <p>In RGMII mode, data bits 3:0 are received on the rising edge of TSEC<sub>n</sub>_RX_CLK.</p> <p>In RTBI mode, TSEC<sub>n</sub>_RXD[3:0] represents RCG[3:0] on the rising edge of TSEC<sub>n</sub>_RX_CLK and RCG[8:5] are received on the falling edge of TSEC<sub>n</sub>_RX_CLK.</p> <p>In TBI mode, TSEC<sub>n</sub>_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals.</p> <p>In RMII mode TSEC<sub>n</sub>_RXD[1:0] represents RXD[1:0], which is considered valid when TSEC<sub>n</sub>_RX_DV (CRS_DV) is asserted, or invalid otherwise. In FIFO mode TSEC<sub>n</sub>_RXD[7:4] with TSEC<sub>n</sub>_RXD[3:0] represent one complete octet of data to be received from the external FIFO device.</p>				
TSEC <sub>n</sub> _RX_ER	I	<p>Receive error</p> <table border="1"> <thead> <tr> <th>State Meaning</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Asserted/Negated</td> <td> <p>In GMII, MII, or RMII mode, if TSEC<sub>n</sub>_RX_ER and TSEC<sub>n</sub>_RX_DV are asserted, the PHY has detected an error in the current frame.</p> <p>In TBI mode, this signal represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.</p> <p>In FIFO mode, this signal represents either receive data error (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols). This signal is not used in the RTBI or RGMII modes.</p> </td> </tr> </tbody> </table>	State Meaning	Description	Asserted/Negated	<p>In GMII, MII, or RMII mode, if TSEC<sub>n</sub>_RX_ER and TSEC<sub>n</sub>_RX_DV are asserted, the PHY has detected an error in the current frame.</p> <p>In TBI mode, this signal represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.</p> <p>In FIFO mode, this signal represents either receive data error (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols). This signal is not used in the RTBI or RGMII modes.</p>
State Meaning	Description					
Asserted/Negated	<p>In GMII, MII, or RMII mode, if TSEC<sub>n</sub>_RX_ER and TSEC<sub>n</sub>_RX_DV are asserted, the PHY has detected an error in the current frame.</p> <p>In TBI mode, this signal represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.</p> <p>In FIFO mode, this signal represents either receive data error (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols). This signal is not used in the RTBI or RGMII modes.</p>					
TSEC <sub>n</sub> _TX_CLK	I	<p>Transmit clock in. In MII mode, TSEC<sub>n</sub>_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC<sub>n</sub>_TX_EN, TSEC<sub>n</sub>_TXD, and TSEC<sub>n</sub>_TX_ER signals. In GMII mode, this signal provides the 2.5 or 25 MHz timing reference during 10Base-T and 100Base-T and comes from the PHY. In 1000Base-T this clock is not used and TSEC<sub>n</sub>_GTX_CLK (125 MHz) becomes the timing reference. The TSEC<sub>n</sub>_GTX_CLK is generated in the eTSEC and provided to the PHY and the MAC. The TSEC<sub>n</sub>_TX_CLK is generated in the PHY and provided to the MAC.</p> <p>In TBI mode, this signal is PMA receive clock 1 at 62.5 MHz, split phase with PMA_RX_CLK0, and is supplied by the SerDes.</p> <p>In RMII mode this signal is the reference clock shared between transmit and receive, and is supplied by the PHY.</p> <p>In FIFO mode the transmit clock is a continuous clock whose maximum is defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and a ratio of 3.2:1 (platform:TxClk) in encoded mode. This signal is not used in the eTSEC RTBI or RGMII modes.</p>				

**Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description
TSEC <sub>n</sub> _TXD[7:4]	O	<p>Transmit data out. In GMII mode, TSEC<sub>n</sub>_TXD[7:0] represents one complete octet of data to be sent from the MAC to the PHY when TSEC_TX_DV is asserted and has no meaning while TSEC<sub>n</sub>_TX_EN is negated.</p> <p>In TBI mode, TSEC<sub>n</sub>_TXD[7:4] represents transmit code group (TCG) bits 7:4. Together, with TCG[9:8] and TCG[3:0], they represent the 10-bit encoded symbol.</p> <p>In GMII or MII mode, TSEC<sub>n</sub>_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSEC<sub>n</sub>_TX_EN is asserted and have no meaning while TSEC<sub>n</sub>_TX_EN is negated.</p> <p>In RGMII or RTBI mode, data bits 3:0 are transmitted on the rising edge of TSEC<sub>n</sub>_TX_CLK, and data bits 7:4 are transmitted on the falling edge of TSEC<sub>n</sub>_TX_CLK.</p> <p>In TBI mode, TSEC<sub>n</sub>_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol.</p> <p>In RMII mode, TSEC<sub>n</sub>_TXD[1:0] represents TXD[1:0], which is valid data sent to the PHY when TSEC<sub>n</sub>_TX_EN is asserted, or undefined otherwise.</p> <p>In FIFO mode, TSEC<sub>n</sub>_TXD[7:4] with TSEC<sub>n</sub>_TXD[3:0] represent one complete octet of data to be received from the external FIFO device.</p> <p>Note that some of these signals are also used during reset to configure the eTSEC interface mode.</p>
TSEC <sub>n</sub> _TX_EN	O	<p>Transmit data valid. In GMII, MII, or RMII mode, if TSEC<sub>n</sub>_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII's or the MII's TSEC<sub>n</sub>_TXD signals.</p> <p>In RGMII mode, TSEC<sub>n</sub>_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this signal on rising and falling edges of the TSEC<sub>n</sub>_GTX_CLK, respectively.</p> <p>In TBI mode, TSEC<sub>n</sub>_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol.</p> <p>In RTBI mode, TSEC<sub>n</sub>_TX_EN represents TCG[4] on the rising edge and TCG[9] on the falling edge of TSEC<sub>n</sub>_GTX_CLK, respectively. Together with TCG[3:0] and TCG[8:5], they represent the 10-bit encoded symbol.</p> <p>In FIFO mode TSEC<sub>n</sub>_TX_EN is used to indicate valid data (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols).</p>
TSEC <sub>n</sub> _TX_ER	O	<p>Transmit error. In GMII or MII mode, assertion of TSEC<sub>n</sub>_TX_ER for one or more clock cycles while TSEC<sub>n</sub>_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC<sub>n</sub>_TX_ER has no effect while operating at 10 Mbps or while TSEC<sub>n</sub>_TX_EN is negated. This signal transitions synchronously with respect to TSEC<sub>n</sub>_TX_CLK.</p> <p>In TBI mode, TSEC<sub>n</sub>_TX_ER represents TCG[9]. Together, with TCG[8:0], they represent the 10-bit encoded symbol.</p> <p>In FIFO mode TSEC<sub>n</sub>_TX_ER represents either transmit data error (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols).</p> <p>This signal is not used in the eTSEC RMII, RTBI, or RGMII modes and is driven low.</p>

## 15.5 Memory Map/Register Definition

The eTSECs use a software model that is a superset of the PowerQUICC III TSEC functionality and is similar to that employed by the Fast Ethernet function supported on the Freescale MPC8260 CPM FCC and in the FEC of the MPC860T.

The eTSEC device is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control, interrupts, and to extract status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always store 0, as writing 1 to reserved bits may have unintended side-effects. Reads from unmapped register addresses return zero. Unless otherwise specified, the read value of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in [Section 15.6.7, “Buffer Descriptors.”](#)

The ten-bit interface (TBI) module MII registers are also described in this section. The TBI registers are defined like PHY registers and, as such, are accessed via the MII management interface in the same way the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) refer to [Section 15.5.4, “Ten-Bit Interface \(TBI\).”](#)

### 15.5.1 Top-Level Module Memory Map

Each of the two eTSECs is allocated 4 Kbytes of memory-mapped space. The space for each eTSEC is divided as indicated in [Table 15-3](#).

**Table 15-3. Module Memory Map Summary**

Address Offset	Function
000–0FF	eTSEC general control/status registers
100–2FF	eTSEC transmit control/status registers
300–4FF	eTSEC receive control/status registers
500–5FF	MAC registers
600–7FF	RMON MIB registers
800–8FF	Hash table registers
900–9FF	—
A00–AFF	FIFO control/status registers
B00–BFF	DMA system registers
C00–C3F	Lossless Flow Control registers
C–FFF	—

### 15.5.2 Detailed Memory Map

[Table 15-4](#) lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for both eTSECs. That is, eTSEC1 starts at 0x2\_4000 address offset and eTSEC3 starts at 0x2\_6000 address offset. The registers for eTSEC1 are listed in [Table 15-4](#), but the registers for the other eTSEC are not. Note that the registers are the same for eTSEC3 except that the offset changes as previously explained and as noted at the end of [Table 15-4](#).

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.

- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 15-4. Module Memory Map**

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
<b>eTSEC General Control and Status Registers</b>				
0x2_4000	TSEC_ID*—Controller ID register	R	0x0124_0000	<a href="#">15.5.3.1.1/15-22</a>
0x2_4004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	<a href="#">15.5.3.1.2/15-23</a>
0x2_4008– 0x2_400C	Reserved	—	—	—
0x2_4010	IEVENT—Interrupt event register	w1c	0x0000_0000	<a href="#">15.5.3.1.3/15-24</a>
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	<a href="#">15.5.3.1.4/15-28</a>
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	<a href="#">15.5.3.1.5/15-30</a>
0x2_401C	Reserved	—	—	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	<a href="#">15.5.3.1.6/15-31</a>
0x2_4024	Reserved	—	—	—
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	<a href="#">15.5.3.1.7/15-33</a>
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	<a href="#">15.5.3.1.8/15-34</a>
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	<a href="#">15.5.3.1.9/15-35</a>
0x2_4034– 0x2_4054	Reserved	—	—	—
<b>eTSEC Transmit Control and Status Registers</b>				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	<a href="#">15.5.3.2.1/15-36</a>
0x2_4104	TSTAT—Transmit status register	w1c	0x0000_0000	<a href="#">15.5.3.2.2/15-38</a>
0x2_4108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	<a href="#">15.5.3.2.3/15-42</a>
0x2_410C	Reserved	—	—	—
0x2_4110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	<a href="#">15.5.3.2.4/15-43</a>
0x2_4114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	<a href="#">15.5.3.2.5/15-44</a>
0x2_4118– 0x2_413C	Reserved	—	—	—
0x2_4140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	<a href="#">15.5.3.2.6/15-44</a>
0x2_4144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	<a href="#">15.5.3.2.7/15-45</a>
0x2_4148– 0x2_417C	Reserved	—	—	—
0x2_4180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	<a href="#">15.5.3.2.8/15-46</a>
0x2_4184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_4188	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_418C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_4190	Reserved	—	—	—
0x2_4194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_4198	Reserved	—	—	—
0x2_419C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_41A0	Reserved	—	—	—
0x2_41A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_41A8	Reserved	—	—	—
0x2_41AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_41B0	Reserved	—	—	—
0x2_41B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_41B8	Reserved	—	—	—
0x2_41BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	<a href="#">15.5.3.2.9/15-46</a>
0x2_41C0– 0x2_41FC	Reserved	—	—	—
0x2_4200	TBASEH—TxBD base address high bits	R/W	0x0000_0000	<a href="#">15.5.3.2.10/15-47</a>
0x2_4204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4208	Reserved	—	—	—
0x2_420C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4210	Reserved	—	—	—
0x2_4214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4218	Reserved	—	—	—
0x2_421C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4220	Reserved	—	—	—
0x2_4224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4228	Reserved	—	—	—
0x2_422C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4230	Reserved	—	—	—
0x2_4234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4238	Reserved	—	—	—
0x2_423C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4240– 0x2_42FC	Reserved	—	—	—
<b>eTSEC Receive Control and Status Registers</b>				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	<a href="#">15.5.3.3.1/15-48</a>

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_4304	RSTAT—Receive status register	w1c	0x0000_0000	15.5.3.3.2/15-50
0x2_4308– 0x2_430C	Reserved	—	—	—
0x2_4310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	15.5.3.3.3/15-52
0x2_4314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	15.5.3.3.4/15-53
0x2_4318– 0x2_432C	Reserved	—	—	—
0x2_4330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	15.5.3.3.5/15-55
0x2_4334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	15.5.3.3.6/15-56
0x2_4338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	15.5.3.3.7/15-57
0x2_433C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	15.5.3.3.8/15-58
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	15.5.3.3.9/15-61
0x2_4344– 0x2_437C	Reserved	—	—	—
0x2_4380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.3.10/15-61
0x2_4384	RBPTR0—RxBd pointer for ring 0	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_4388	Reserved	—	—	—
0x2_438C	RBPTR1*—RxBd pointer for ring 1	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_4390	Reserved	—	—	—
0x2_4394	RBPTR2*—RxBd pointer for ring 2	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_4398	Reserved	—	—	—
0x2_439C	RBPTR3*—RxBd pointer for ring 3	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_43A0	Reserved	—	—	—
0x2_43A4	RBPTR4*—RxBd pointer for ring 4	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_43A8	Reserved	—	—	—
0x2_43AC	RBPTR5*—RxBd pointer for ring 5	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_43B0	Reserved	—	—	—
0x2_43B4	RBPTR6*—RxBd pointer for ring 6	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_43B8	Reserved	—	—	—
0x2_43BC	RBPTR7*—RxBd pointer for ring 7	R/W	0x0000_0000	15.5.3.3.11/15-62
0x2_43C0– 0x2_43FC	Reserved	—	—	—
0x2_4400	RBASEH—RxBd base address high bits	R/W	0x0000_0000	15.5.3.3.12/15-63
0x2_4404	RBASE0—RxBd base address of ring 0	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4408	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_440C	RBASE1*—RxB D base address of ring 1	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4410	Reserved	—	—	—
0x2_4414	RBASE2*—RxB D base address of ring 2	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4418	Reserved	—	—	—
0x2_441C	RBASE3*—RxB D base address of ring 3	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4420	Reserved	—	—	—
0x2_4424	RBASE4*—RxB D base address of ring 4	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4428	Reserved	—	—	—
0x2_442C	RBASE5*—RxB D base address of ring 5	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4430	Reserved	—	—	—
0x2_4434	RBASE6*—RxB D base address of ring 6	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4438	Reserved	—	—	—
0x2_443C	RBASE7*—RxB D base address of ring 7	R/W	0x0000_0000	15.5.3.3.13/15-63
0x2_4440– 0x2_44FC	Reserved	—	—	—
<b>eTSEC MAC Registers</b>				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	15.5.3.5.1/15-66
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	15.5.3.5.2/15-68
0x2_4508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	15.5.3.5.3/15-70
0x2_450C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	15.5.3.5.4/15-71
0x2_4510	MAXFRM—Maximum frame length	R/W	0x0000_0600	15.5.3.5.5/15-72
0x2_4514– 0x2_451C	Reserved	—	—	—
0x2_4520	MIIMCFG—MII management configuration	R/W	0x0000_0007	15.5.3.5.6/15-72
0x2_4524	MIIMCOM—MII management command	R/W	0x0000_0000	15.5.3.5.7/15-73
0x2_4528	MIIMADD—MII management address	R/W	0x0000_0000	15.5.3.5.8/15-74
0x2_452C	MIIMCON—MII management control	WO	0x0000_0000	15.5.3.5.9/15-74
0x2_4530	MIIMSTAT—MII management status	R	0x0000_0000	15.5.3.5.10/15-75
0x2_4534	MIIMIND—MII management indicator	R	0x0000_0000	15.5.3.5.11/15-75
0x2_4538	Reserved	—	—	—
0x2_453C	IFSTAT—Interface status	R	0x0000_0000	15.5.3.5.12/15-76
0x2_4540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	15.5.3.5.13/15-76
0x2_4544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	15.5.3.5.14/15-77

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_4548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	15.5.3.5.15/15-78 15.5.3.5.16/15-78
0x2_454C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x2_4550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x2_4554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x2_4558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x2_455C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x2_4560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x2_4564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x2_4568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x2_456C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x2_4570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x2_4574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x2_4578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x2_457C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x2_4580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x2_4584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x2_4588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x2_458C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x2_4590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	
0x2_4594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	
0x2_4598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	15.5.3.5.15/15-78 15.5.3.5.16/15-78
0x2_459C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x2_45A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x2_45A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x2_45A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x2_45AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x2_45B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x2_45B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x2_45B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x2_45BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
0x2_45C0– 0x2_467C	Reserved	—	—	—
<b>eTSEC Transmit and Receive Counters</b>				
0x2_4680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	15.5.3.6.1/15-79
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	15.5.3.6.2/15-80
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	15.5.3.6.3/15-80



Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.4/15-81</a>
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.5/15-81</a>
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.6/15-82</a>
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	<a href="#">15.5.3.6.7/15-82</a>
<b>eTSEC Receive Counters</b>				
0x2_469C	RBYT—Receive byte counter	R/W	0x0000_0000	<a href="#">15.5.3.6.8/15-83</a>
0x2_46A0	RPKT—Receive packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.9/15-83</a>
0x2_46A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.10/15-84</a>
0x2_46A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.11/15-84</a>
0x2_46AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.12/15-85</a>
0x2_46B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.13/15-85</a>
0x2_46B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.14/15-86</a>
0x2_46B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	<a href="#">15.5.3.6.15/15-86</a>
0x2_46BC	RALN—Receive alignment error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.16/15-87</a>
0x2_46C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.17/15-87</a>
0x2_46C4	RCDE—Receive code error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.18/15-88</a>
0x2_46C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.19/15-88</a>
0x2_46CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.20/15-89</a>
0x2_46D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.21/15-89</a>
0x2_46D4	RFRG—Receive fragments counter	R/W	0x0000_0000	<a href="#">15.5.3.6.22/15-90</a>
0x2_46D8	RJBR—Receive jabber counter	R/W	0x0000_0000	<a href="#">15.5.3.6.23/15-90</a>
0x2_46DC	RDRP—Receive drop counter	R/W	0x0000_0000	<a href="#">15.5.3.6.24/15-91</a>
<b>eTSEC Transmit Counters</b>				
0x2_46E0	TBYT—Transmit byte counter	R/W	0x0000_0000	<a href="#">15.5.3.6.25/15-91</a>
0x2_46E4	TPKT—Transmit packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.26/15-92</a>
0x2_46E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.27/15-92</a>
0x2_46EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.28/15-93</a>
0x2_46F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.29/15-93</a>
0x2_46F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.30/15-94</a>
0x2_46F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.31/15-94</a>
0x2_46FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.32/15-95</a>
0x2_4700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.33/15-95</a>
0x2_4704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.34/15-96</a>

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_4708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	15.5.3.6.35/15-96
0x2_470C	TNCL—Transmit total collision counter	R/W	0x0000_0000	15.5.3.6.36/15-97
0x2_4710	Reserved	—	—	—
0x2_4714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	15.5.3.6.37/15-97
0x2_4718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	15.5.3.6.38/15-98
0x2_471C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	15.5.3.6.39/15-98
0x2_4720	TXCF—Transmit control frame counter	R/W	0x0000_0000	15.5.3.6.40/15-99
0x2_4724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	15.5.3.6.41/15-99
0x2_4728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	15.5.3.6.42/15-100
0x2_472C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	15.5.3.6.43/15-100
<b>eTSEC Counter Control and TOE Statistics Registers</b>				
0x2_4730	CAR1—Carry register one register <sup>3</sup>	R	0x0000_0000	15.5.3.6.44/15-100
0x2_4734	CAR2—Carry register two register <sup>3</sup>	R	0x0000_0000	15.5.3.6.45/15-103
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	15.5.3.6.46/15-104
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	15.5.3.6.47/15-105
0x2_4740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	15.5.3.6.48/15-106
0x2_4744– 0x2_47FC	Reserved	—	—	—
<b>Hash Function Registers</b>				
0x2_4800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	15.5.3.7.1/15-107
0x2_4804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x2_4808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x2_480C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x2_4810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x2_4814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x2_4818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x2_481C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	
0x2_4820– 0x2_487C	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	15.5.3.7.2/15-108
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0– 0x2_49FC	Reserved	—	—	—
<b>eTSEC FIFO Control Registers</b>				
0x2_4A00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_00C0	15.5.3.8.1/15-109
0x2_4A04– 0x2_4AFC	Reserved	—	—	—
<b>eTSEC DMA Attribute Registers</b>				
0x2_4B00– 0x2_4BF4	Reserved	—	—	—
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	15.5.3.9.1/15-110
0x2_4BFC	ATTRELI—Attribute extract length and extract index register	R/W	0x0000_0000	15.5.3.9.2/15-111
<b>eTSEC Lossless Flow Control Registers</b>				
0x2_4C00	RQPRM0*—Receive Queue Parameters register 0	R/W	0x0000_0000	15.5.3.10.1/15-112
0x2_4C04	RQPRM1*—Receive Queue Parameters register 1	R/W	0x0000_0000	
0x2_4C08	RQPRM2*—Receive Queue Parameters register 2	R/W	0x0000_0000	
0x2_4C0C	RQPRM3*—Receive Queue Parameters register 3	R/W	0x0000_0000	
0x2_4C10	RQPRM4*—Receive Queue Parameters register 4	R/W	0x0000_0000	
0x2_4C14	RQPRM5*—Receive Queue Parameters register 5	R/W	0x0000_0000	
0x2_4C18	RQPRM6*—Receive Queue Parameters register 6	R/W	0x0000_0000	
0x2_4C1C	RQPRM7*—Receive Queue Parameters register 7	R/W	0x0000_0000	
0x2_4C20– 0x2_4C40	Reserved	—	—	—
0x2_4C44	RFBPTR0*—Last Free RxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.10.2/15-113
0x2_4C48	Reserved	—	—	—
0x2_4C4C	RFBPTR1*—Last Free RxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.10.2/15-113
0x2_4C50	Reserved	—	—	—
0x2_4C54	RFBPTR2*—Last Free RxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.10.2/15-113

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name <sup>1</sup>	Access <sup>2</sup>	Reset	Section/Page
0x2_4C58	Reserved	—	—	—
0x2_4C5C	RFBPTR3*—Last Free RxBD pointer for ring 3	R/W	0x0000_0000	<a href="#">15.5.3.10.2/15-113</a>
0x2_4C60	Reserved	—	—	—
0x2_4C64	RFBPTR4*—Last Free RxBD pointer for ring 4	R/W	0x0000_0000	<a href="#">15.5.3.10.2/15-113</a>
0x2_4C68	Reserved	—	—	—
0x2_4C6C	RFBPTR5*—Last Free RxBD pointer for ring 5	R/W	0x0000_0000	<a href="#">15.5.3.10.2/15-113</a>
0x2_4C70	Reserved	—	—	—
0x2_4C74	RFBPTR6*—Last Free RxBD pointer for ring 6	R/W	0x0000_0000	<a href="#">15.5.3.10.2/15-113</a>
0x2_4C78	Reserved	—	—	—
0x2_4C7C	RFBPTR7*—Last Free RxBD pointer for ring 7	R/W	0x0000_0000	<a href="#">15.5.3.10.2/15-113</a>
<b>eTSEC Future Expansion Space</b>				
0x2_4C00– 0x2_4D94	Reserved	—	—	—
<b>eTSEC3</b>				
0x2_6000– 0x2_6FFF	eTSEC3 REGISTERS <sup>4</sup>			

<sup>1</sup> Registers denoted \* are new to the enhanced TSEC and not supported by PowerQUICC III TSECs.

<sup>2</sup> Key: R = read only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing.

<sup>3</sup> Cleared on read.

<sup>4</sup> eTSEC3 has the same memory-mapped registers that are described for eTSEC1 from 0x 2\_4000 to 0x2\_4FFF, except the offsets are from 0x 2\_6000 to 0x2\_6FFF.

## 15.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the eTSEC registers. Because all of the eTSEC registers are 32 bits wide, only 32-bit register accesses are supported.

### 15.5.3.1 eTSEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

#### 15.5.3.1.1 Controller ID Register (TSEC\_ID)

The controller ID register (TSEC\_ID) is a read-only register. The TSEC\_ID register is used to identify the eTSEC block and revision.

Offset eTSEC1:0x2\_4000; eTSEC3:0x2\_6000

Access: Read only

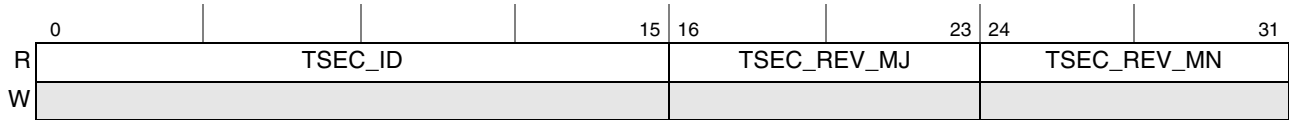


Figure 15-2. TSEC\_ID Register

Table 15-9 describes the fields of the TSEC\_ID register.

Table 15-5. TSEC\_ID Field Descriptions

Bits	Name	Description
0–15	TSEC_ID	Value identifying the eTSEC (10/100/1000 Ethernet MAC). 0124 Unique identifier for eTSEC with 8 Rx and 8 Tx BD rings.
16–23	TSEC_REV_MJ	Value identifies the major revision of the eTSEC. 00 Initial revision
24–31	TSEC_REV_MN	Value identifies the minor revision of the eTSEC.

### 15.5.3.1.2 Controller ID Register (TSEC\_ID2)

The controller ID register (TSEC\_ID2) is a read-only register. The TSEC\_ID2 register is used to identify the eTSEC block configuration.

Offset eTSEC1:0x2\_4004; eTSEC3:0x2\_6004

Access: Read only

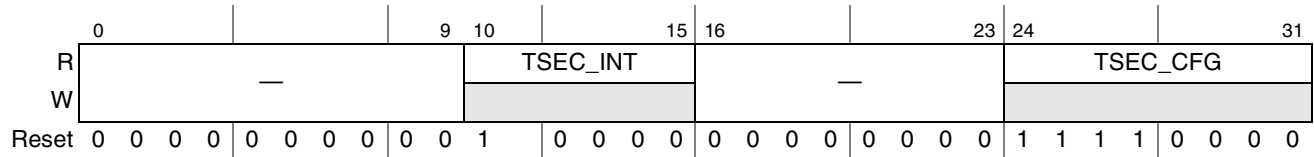


Figure 15-3. TSEC\_ID2 Register

Table 15-6 describes the fields of the TSEC\_ID2 register.

**Table 15-6. TSEC\_ID2 Field Descriptions**

Bits	Name	Description														
0–9	—	Reserved														
10–15	TSEC_INT	Interface mode support. <table border="1" data-bbox="529 436 1349 869"> <thead> <tr> <th>Bit</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>0 Ethernet mode not supported 1 Ethernet mode supported</td> </tr> <tr> <td>11</td> <td>0 FIFO mode not supported 1 FIFO mode supported</td> </tr> <tr> <td>12</td> <td>Reserved</td> </tr> <tr> <td>13</td> <td>0 Can be configured to run in FIFO 8-bit mode 1 FIFO 8-bit mode off</td> </tr> <tr> <td>14</td> <td>0 Can be configured to run in Ethernet normal/full mode 1 Ethernet normal/full mode off</td> </tr> <tr> <td>15</td> <td>0 Can be configured to run in Ethernet reduced mode 1 Ethernet reduced mode off</td> </tr> </tbody> </table>	Bit	Mode	10	0 Ethernet mode not supported 1 Ethernet mode supported	11	0 FIFO mode not supported 1 FIFO mode supported	12	Reserved	13	0 Can be configured to run in FIFO 8-bit mode 1 FIFO 8-bit mode off	14	0 Can be configured to run in Ethernet normal/full mode 1 Ethernet normal/full mode off	15	0 Can be configured to run in Ethernet reduced mode 1 Ethernet reduced mode off
Bit	Mode															
10	0 Ethernet mode not supported 1 Ethernet mode supported															
11	0 FIFO mode not supported 1 FIFO mode supported															
12	Reserved															
13	0 Can be configured to run in FIFO 8-bit mode 1 FIFO 8-bit mode off															
14	0 Can be configured to run in Ethernet normal/full mode 1 Ethernet normal/full mode off															
15	0 Can be configured to run in Ethernet reduced mode 1 Ethernet reduced mode off															
16–23	—	Reserved														
24–31	TSEC_CFG	Value identifies configuration options of the eTSEC. 00 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are off F0 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are on 30 eTSEC multiple ring support is OFF and Rx TOE, Filer and Tx TOE supports are on 50 eTSEC multiple ring and filer supports are OFF and Rx TOE and Tx TOE supports are on														

### 15.5.3.1.3 Interrupt Event Register (IEVENT)

Interrupt events cause bits in the IEVENT register to be set. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the interrupt mask register (IMASK), the event also causes a hardware interrupt at the PIC. A bit in the interrupt event register is cleared by writing a 1 to that bit position. A write of 0 has no effect.

Each eTSEC can issue three kinds of hardware interrupt to the PIC:

1. Transmit interrupts—Issued whenever bits TXB or TXF of IEVENT are set to 1 and either transmit interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for TXF. To negate this hardware interrupt, software must clear both TXB and TXF bits.
2. Receive interrupts—Issued whenever bits RXB or RXF of IEVENT are set to 1 and either receive interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for RXF. To negate this hardware interrupt, software must clear both RXB and RXF bits.
3. Error and diagnostic interrupts—Issued whenever bits MAG, GTSC, GRSC, TXC, RXC, BABR, BABT, LC, CRL, FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN or BSY of IEVENT are set to 1. Software must clear all of these bits to negate an error/diagnostic hardware interrupt.
  - Magic Packet reception event is: MAG

- Operational diagnostics are events on: GTSC, GRSC, TXC and RXC
- Interrupts resulting from errors/problems detected in the network or transceiver are: BABR, BABT, LC and CRL
- Interrupts resulting from internal errors are: FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN and BSY

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Figure 15-4 describes the definition for the IEVENT register.

Offset eTSEC1:0x2\_4010; eTSEC3: 0x2\_6010 Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	—	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	—	LC	CRL	XFUN
W	w1c	w1c	w1c	w1c	—	w1c	w1c	w1c	w1c	w1c	w1c	w1c	—	w1c	w1c	w1c
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXB	—			MAG	MMRD	MMWR	GRSC	RXF	—			FIR	FIQ	DPE	PERR
W	w1c	—			w1c	w1c	w1c	w1c	w1c	—			w1c	w1c	w1c	w1c
Reset	All zeros															

**Figure 15-4. IEVENT Register Definition**

Table 15-7 describes the fields of the IEVENT register.

**Table 15-7. IEVENT Field Descriptions**

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received. 1 Excessive frame received.
1	RXC	Receive control interrupt. A control frame was received while MACCFG1[Rx_Flow] is set. As soon as the transmitter finishes sending the current frame, a pause operation is performed. 0 Control frame not received. 1 Control frame received.
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. 0 No frame received and discarded. 1 Frame received and discarded.
3	EBERR	Internal bus error. This bit indicates that a system bus error occurred while a DMA transaction was underway. As a result, transferred data is expected to be partially or completely invalid. 0 No system bus error occurred. 1 System bus error occurred.
4	—	Reserved
5	MSRO	MIB counter overflow. This interrupt is asserted if the count for one of the MIB counters has exceeded the size of its register. 0 MIB count not exceeding its register size. 1 MIB count exceeds its register size.

Table 15-7. IEVENT Field Descriptions (continued)

Bits	Name	Description
6	GTSC	Graceful transmit stop complete. This interrupt is asserted for one of two reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. <ul style="list-style-type: none"> <li>• A graceful stop, which was initiated by setting DMACTRL[GTS], is now complete.</li> <li>• A transmission of a flow control PAUSE frame, which was initiated by setting TCTRL[TFC_PAUSE], is now complete.</li> </ul> 0 No graceful stop interrupt. 1 Graceful stop requested.
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's maximum frame length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. 0 Transmitted frame length not exceeding maximum frame length. 1 Transmitted frame length exceeding maximum frame length when MACCFG2[Huge Frame] = 0.
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted. 1 Control frame transmitted.
9	TXE	Transmit error. This bit indicates that an error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by the eTSEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL, XFUN). 0 No transmit channel error occurred. 1 Transmit channel error occurred.
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated. 1 Transmit buffer descriptor updated.
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (interrupt) bit in the status word of the buffer descriptor is set. The specific transmit queue that was updated has its TXF bit set in TSTAT. 0 No frame transmitted/TxBD not updated. 1 Frame transmitted/TxBD updated.
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred.
14	CRL	Collision retry limit. This bit indicates that the number of successive transmission collisions has exceeded the MAC's half-duplex register's retransmission maximum count (HAFDUP[Retransmission Maximum]). The frame is discarded without being transmitted and transmission of the next frame commences. This only occurs while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum. 1 Successive transmission collisions exceed maximum.
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun. 1 Transmit FIFO underrun.
16	RXB	Receive buffer. This bit indicates that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated. 1 Receiver buffer descriptor updated.



Table 15-7. IEVENT Field Descriptions (continued)

Bits	Name	Description
17–19	—	Reserved
20	MAG	Magic Packet detected when the eTSEC is in Magic Packet detection mode (MACCFG2[MPEN] = 1). 0 No Magic Packet received, or Magic Packet mode was not enabled. 1 A Magic Packet was received while in Magic Packet mode. MACCFG2[MPEN] is also cleared upon receiving the Magic Packet.
21	MMRD	MII management read completion 0 MII management read not issued or in process. 1 MII management read completed that was initiated by a user through the MII Scan or Read cycle command.
22	MMWR	MII management write completion 0 MII management write not issued or in process. 1 MII management write completed that was initiated by a user write to the MIIMCON register.
23	GRSC	Graceful receive stop complete. This interrupt is asserted if a graceful receive stop is completed. It allows the user to know if the system has completed the stop and it is safe to write to receive registers (status, control or configuration registers) that are used by the system during normal operation. 0 Graceful stop not completed. 1 Graceful stop completed.
24	RXF	Receive frame interrupt. This bit indicates that a frame was received and the last receive buffer descriptor (RxBd) in that frame was updated. This occurs either if the I (interrupt) bit in the buffer descriptor status word is set, or an overrun error occurs. The specific receive queue that was updated has its RXF bit set in RSTAT. 0 Frame not received. 1 Frame received.
25–27	—	Reserved
28	FIR	The receive queue filer result is invalid, either because not enough time between frames was available to find a matching rule, or no entry in the filer table could be matched. 0 Receive queue filer reached a definite result; however, bit FIQ may still be set if a frame was filed to a disabled RxBd ring. 1 Receive queue filer was unable to reach a definite result. In this case, bit FIQ will also be set if no entry in the filer table could provide a rule match.
29	FIQ	Filed frame to invalid receive queue. This bit indicates that either the receive queue filer chose to DMA a received frame to a disabled RxBd ring, or that no rule in the filer table could be matched. 0 Received frames filed to valid queues or rejected. Note that a frame may be rejected if the filer has insufficient time to reach a conclusive result between frames, in which case bit FIR is set. 1 Received frames filed to RxBd rings that are not enabled. The frame is discarded. If bit FIR is also set this indicates that the filer exhausted all of its table entries without a rule match.
30	DPE	Internal data parity error. This bit indicates that the eTSEC has detected a parity error on its stored data, which is likely to compromise the validity of recently transferred frames. 0 No parity errors detected. 1 Data held in the FIFO or filer arrays is expected to be corrupted due to a parity error.
31	PERR	Receive frame parse error for TCP/IP off-load. This bit indicates that a received frame could not be parsed unambiguously, due to encapsulated header type fields contradicting each other. 0 Received frame parsed successfully. 1 Received frame parse revealed header inconsistencies.

### 15.5.3.1.4 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events in the IEVENT register are permitted to participate in generating hardware interrupts to the PIC. All implemented bits in this register are R/W and cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, the PIC will receive an interrupt (for each eTSEC these are grouped into transmit, receive, and error/diagnostic interrupts). The interrupt signal remains asserted until either the IEVENT bit is cleared, by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Figure 15-5 describes the IMASK register.

Offset eTSEC1:0x2\_4014; eTSEC3:0x2\_6014

Access: Read/Write

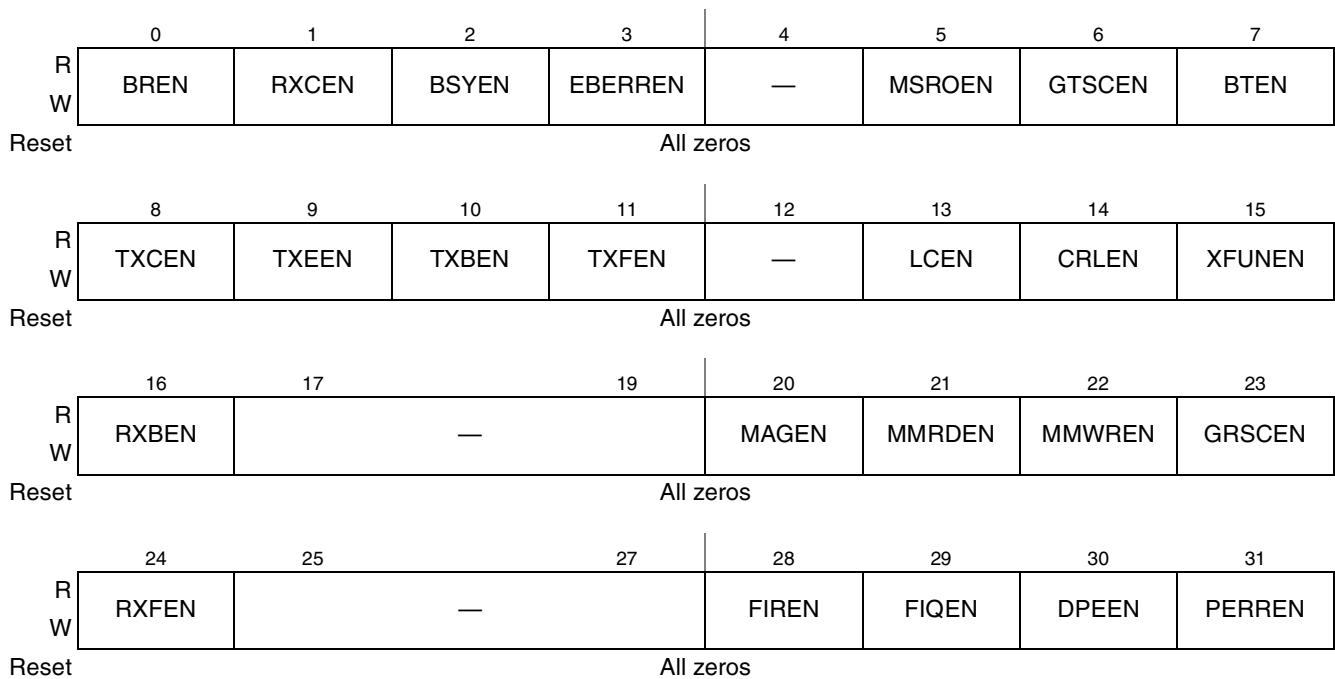


Figure 15-5. IMASK Register Definition

Table 15-8 describes the fields of the IMASK register.

Table 15-8. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MIB counter overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable

Table 15-8. IMASK Field Descriptions (continued)

Bits	Name	Description
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRLLEN	Collision retry limit enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–19	—	Reserved
20	MAGEN	Magic Packet received interrupt enable
21	MMRDEN	MII management read completion interrupt enable
22	MMWREN	MII management write completion interrupt enable
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–27	—	Reserved
28	FIREN	Filer invalid result interrupt enable
29	FIQEN	Filed frame to invalid queue interrupt enable
30	DPEEN	Data parity error interrupt enable
31	PERREN	Receive frame parse error enable

### 15.5.3.1.5 Error Disabled Register (EDIS)

Figure 15-6 describes the definition for the EDIS register. The error disabled register allows the user to disable an error interruption, possibly to avoid spurious error indications external to the eTSECs.

Offset eTSEC1:0x2\_4018; eTSEC3:0x2\_6018

Access: Read/Write

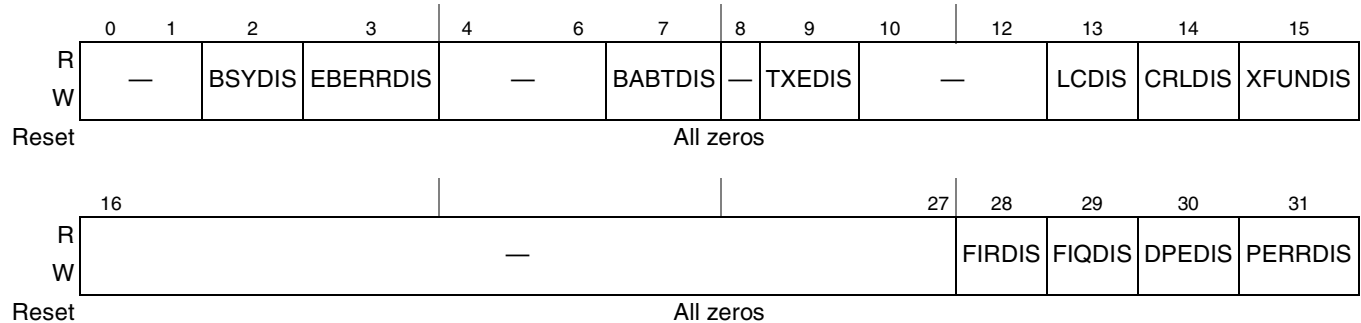


Figure 15-6. EDIS Register Definition

Table 15-9 describes the fields of the EDIS register.

Table 15-9. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable. 0 Allow eTSEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable. 0 Allow eTSEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–6	—	Reserved
7	BABTDIS	Babbling transmit error disable. 0 Allow eTSEC to report IEVENT[BABT] status and set the buffer descriptor TR field. 1 Do not set IEVENT[BABT] nor the buffer descriptor TR field.
8	—	Reserved
9	TXEDIS	Transmit error disable. 0 Allow eTSEC to report IEVENT[TXE] status. 1 Do not set IEVENT[TXE] if TXE condition occurs.
10–12	—	Reserved
13	LCDIS	Late collision disable. 0 Allow eTSEC to report IEVENT[LC] status, set the buffer descriptor LC field, and halt buffer descriptor queue if LC condition occurs. 1 Do not set IEVENT[LC] nor the buffer descriptor LC field, and do not halt buffer descriptor queue if LC condition occurs.

Table 15-9. EDIS Field Descriptions (continued)

Bits	Name	Description
14	CRLDIS	Collision retry limit disable. 0 Allow eTSEC to report IEVENT[CRL] status, set the buffer descriptor RL field, and halt buffer descriptor queue if CRL condition occurs. 1 Do not set IEVENT[CRL] nor the buffer descriptor RL field, and do not halt buffer descriptor queue if CRL condition occurs.
15	XFUNDIS	Transmit FIFO underrun disable. 0 Allow eTSEC to report IEVENT[XFUN] status, set the buffer descriptor UN field, and halt buffer descriptor queue if XFUN condition occurs. 1 Do not set IEVENT[XFUN] nor the buffer descriptor UN field, and do not halt buffer descriptor queue if XFUN condition occurs.
16–27	—	Reserved
28	FIRDIS	Filer invalid result error disable. 0 Allow eTSEC to report IEVENT[FIR] status. 1 Do not set IEVENT[FIR] if eTSEC fails to reach a definite filer result when attempting to file a received frame, but discard the frame silently.
29	FIQDIS	Filed frame to invalid queue error disable. 0 Allow eTSEC to report IEVENT[FIQ] status. 1 Do not set IEVENT[FIQ] if eTSEC attempts to file a received frame to an invalid (disabled) RxBD ring, but discard the frame silently.
30	DPEDIS	Data parity error disable. 0 Allow eTSEC to report IEVENT[DPE] status. 1 Do not set IEVENT[DPE] if a parity error occurs in eTSEC's FIFO or filer arrays.
31	PERRDIS	Receive frame parse error disable. 0 Allow eTSEC to report IEVENT[PERR] status. 1 Do not set IEVENT[PERR] if a parse error occurs on a received frame.

### 15.5.3.1.6 Ethernet Control Register (ECNTRL)

ECNTRL is a register writable by the user to reset, configure, and initialize the eTSEC. Note that the FIFM, GMIIM, TBIM, RPM, and RMM fields are read-only, having been set after sampling signals at power-on-reset.

Figure 15-7 describes the definition for the ECNTRL register.

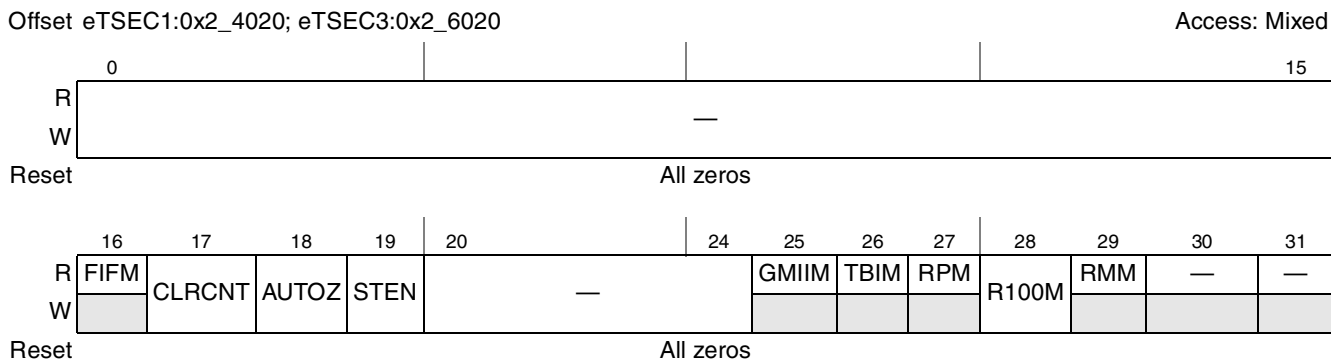


Figure 15-7. ECNTRL Register Definition

Table 15-10 describes the fields of the ECNTRL register.

**Table 15-10. ECNTRL Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16	FIFM	FIFO mode enable. If this bit is set, 8-bit FIFO interface mode is enabled. This bit can be pin configured at reset to set or clear. See <a href="#">Section 4.4.3, “Power-On Reset Configuration.”</a> 0 Interface to external signals via the Ethernet MAC. 1 Interface to external signals via the 8-bit FIFO interface, bypassing the Ethernet MAC. Frame parsing in this mode automatically assumes that IP packets are being received and transmitted. See FIFOCFG register for configuration of the FIFO interface.
17	CLRCNT	Clear all statistics counters 0 Allow MIB counters to continue to increment. 1 Reset all MIB counters. This bit is self-resetting.
18	AUTOZ	Automatically zero MIB counter values. 0 The user must write the addressed counter zero after a host read. 1 The addressed counter value is automatically cleared to zero after a host read. This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
19	STEN	MIB counter statistics enabled. 0 Statistics not enabled 1 Enables internal counters to update This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
20–24	—	Reserved
25	GMIIM	GMII interface mode. If this bit is set, a PHY with a GMII or RGMII interface is expected to be connected. If cleared, a PHY with an MII or RMII interface is expected. The user should then set MACCFG2[I/F Mode] accordingly. The state of this status bit is defined during power-on reset. See <a href="#">Section 4.4.3, “Power-On Reset Configuration.”</a> 0 MII or RMII mode interface expected 1 GMII or RGMII mode interface expected
26	TBIM	Ten-bit interface mode. If this bit is set, ten-bit interface mode is enabled. This bit can be pin-configured at reset to set or clear. See <a href="#">Section 4.4.3, “Power-On Reset Configuration.”</a> 0 GMII or MII or RMII mode interface 1 TBI mode interface
27	RPM	Reduced-pin mode for Gigabit interfaces. If this bit is set, a reduced-pin interface is expected on either Ethernet and FIFO interfaces. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See <a href="#">Section 4.4.3, “Power-On Reset Configuration.”</a> 0 GMII or MII or TBI in non-reduced-pin mode configuration 1 RGMII or RTBI reduced-pin mode FIFO configured for 8-bit operation
28	R100M	RGMII/RMII 100 mode. This bit is ignored unless RPM or RMM are set and MACCFG2[I/F Mode] is assigned to 10/100 (01). If this bit is set, the eTSEC interface is in 100 Mbps speed. 0 RGMII is in 10 Mbps mode; RMII is in 10 Mbps mode, and every 10th RMII Reference clock is used to transfer data 1 RGMII is in 100 Mbps mode; RMII is in 100 Mbps mode, and data is transferred on every Reference clock

Table 15-10. ECNTRL Field Descriptions (continued)

Bits	Name	Description
29	RMM	Reduced-pin mode for 10/100 interfaces. If this bit is set, a RMII pin interface is expected. Valid only if FIFM = 0 and TBIM = 0. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See <a href="#">Section 4.4.3, “Power-On Reset Configuration.”</a> 0 MII, RGMII, GMII, TBI, or RTBI mode configuration 1 RMII mode
30–31	—	Reserved

The different interface configurations indicated by registers ECNTRL and MACCFG2 are summarized in [Table 15-11](#).

Table 15-11. eTSEC Interface Configurations

Interface Mode	ECNTRL Field						MACCFG2 Field
	FIFM	GMIIM	TBIM	RPM	R100M	RMM	I/F Mode
FIFO 8-bits	1	0	0	1	—	0	—
TBI 1 Gbps	0	0	1	0	—	0	10
RTBI 1 Gbps	0	0	1	1	—	0	10
GMII 1 Gbps	0	1	0	0	—	0	10
RGMII 1 Gbps	0	1	0	1	—	—	10
RGMII 100 Mbps	0	1	0	1	1	—	01
RGMII 10 Mbps	0	1	0	1	0	0	01
MII 10/100 Mbps	0	0	0	0	—	0	01
RMII 100 Mbps	0	0	0	0	1	1	01
RMII 10 Mbps	0	0	0	0	0	1	01

### 15.5.3.1.7 Pause Time Value Register (PTV)

PTV is a 32-bit register written by the user to store the pause duration used when the eTSEC initiates an IEEE 802.3 PAUSE control frame via TCTRL[TFC\_PAUSE]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of *pause\_quanta*, equal to 512 bit times. The pause time can range from 0 to 65,535 *pause\_quanta*, or 0 to 33,553,920 bit times. See [Section 15.6.3.9, “Flow Control,”](#) for additional details. [Figure 15-8](#) describes the definition for the PTV register.

Offset eTSEC1:0x2\_4028; eTSEC3:0x2\_6028

Access: Read/Write

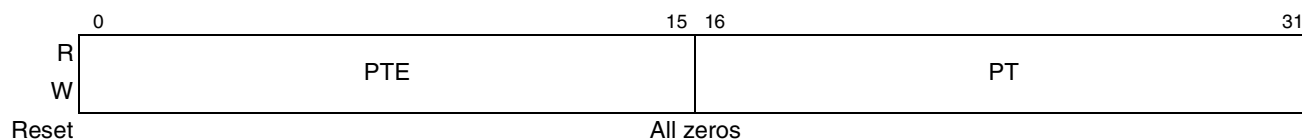


Figure 15-8. PTV Register Definition

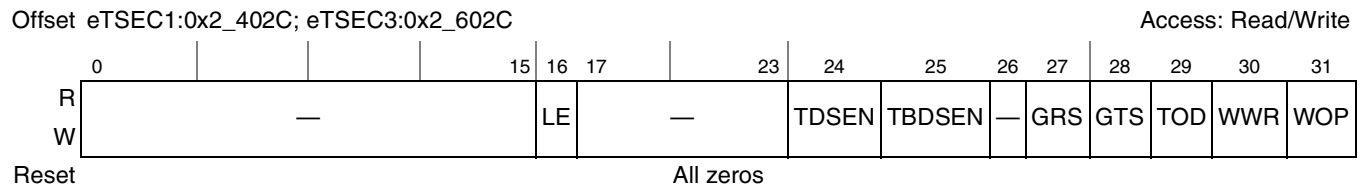
Table 15-12 describes the fields of the PTV register.

**Table 15-12. PTV Field Descriptions**

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 PAUSE frame format requires this parameter to be cleared.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (i.e. 512 bit times). This pause value is used as part of the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. See <a href="#">Section 15.6.3.9, “Flow Control,”</a> on <a href="#">page 15-151</a> for more information.

### 15.5.3.1.8 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. [Figure 15-9](#) describes the definition for the DMACTRL register.



**Figure 15-9. DMACTRL Register**

Table 15-13 describes the fields of the DMACTRL register.

**Table 15-13. DMACTRL Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16	LE	Little-endian descriptor mode enable. This bit controls both the reading and writing of descriptors; data buffers are always transferred in network byte order. 0 RxBDs and TxBDs are interpreted with big-endian byte ordering, as shown in <a href="#">Section 15.6.7.1, “Data Buffer Descriptors.”</a> 1 RxBDs and TxBDs are interpreted with little-endian byte ordering. That is, the 16 bits of flags are considered a complete half-word unit, the buffer length is considered another complete half-word unit, and the buffer pointer is considered a complete word unit.
17–23	—	Reserved
24	TDSEN	Tx Data snoop enable. 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSSEN	TxBD snoop enable. 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved



**Table 15-13. DMACTRL Field Descriptions (continued)**

Bits	Name	Description
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received. (That is, after a valid end of frame was received). The contents of the Rx FIFO are then written to memory, and the IEVENT[GRSC] is set to indicate that all current receive buffers have been closed. Because the receive enable bit of the MAC may still be set, the MAC may continue to receive but the eTSEC ignores the receive data until GRS is cleared. If this bit is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxB. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the eTSEC hardware without fear of conflict. 0 eTSEC scans input data stream for valid frame. 1 eTSEC stops receiving frames following completion of current frame.
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after all frames that are currently in the Tx FIFO or scheduled have been transmitted, and the GTSC interrupt in the IEVENT register is asserted. A frame that has started reading buffer descriptors or data from memory will be read to completion and transmitted before the GTSC interrupt occurs. However, if no frame has been scheduled for transmission and the Tx FIFO is empty, the GTSC interrupt is asserted immediately. Once transmission has completed, clearing GTS will “restart” transmit. 0 Controller continues. 1 Controller stops transmission after completion of current frame.
29	TOD	Transmit on demand for TxBD ring 0. This bit is applicable only to the transmitter, and requires both TCTRL[TXSCHED] = 00 and DMACTRL[WOP] = 0. If 1 is written to this bit, the eTSEC immediately begins fetching the next TxBD from ring 0, avoiding waiting the normal polling time to check the TxBD's R bit. This bit is always read as 0. 0 eTSEC continues waiting for the TxBD ring 0 poll timer to expire. 1 eTSEC immediately fetches a new TxBD from ring 0, and resets the poll timer.
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, XFUN, LC, CRL, RXB, RXF, the eTSEC waits for acknowledgement from system that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll for TxBD ring 0. This bit, which is applicable only to the transmitter and when TCTRL[TXSCHED] = 00, provides the user the option for the eTSEC to periodically poll TxBDs or to wait for software to tell eTSEC to fetch a buffer descriptor. While operating in the “Wait” mode, the eTSEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven. To resume transmission, software must clear TSTAT[THLT]. 0 Poll TxBD on ring 0 every 512 serial clocks. 1 Do not poll, but wait for TSTAT[THLT] to be cleared by the user.

### 15.5.3.1.9 TBI Physical Address Register (TBIPA)

The TBIPA, shown in [Figure 15-10](#), is writable by the user to assign a physical address to the TBI for MII management configuration. The TBI registers are accessed at the offset of TBIPA. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to [Section 15.5.4, “Ten-Bit Interface \(TBI\).”](#)

Offset eTSEC1:0x2\_4030; eTSEC3:0x2\_6030

Access: Read/Write

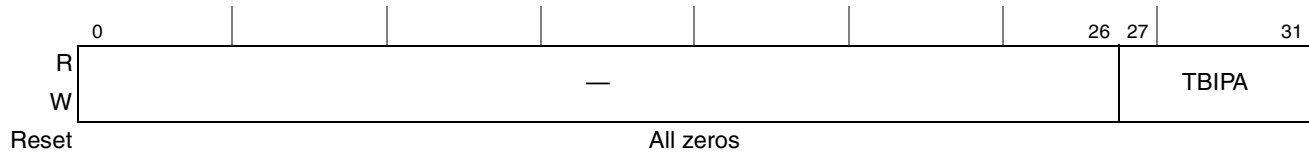


Figure 15-10. TBIPA Register Definition

Table 15-14 describes the fields of the TBIPA register.

Table 15-14. TBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	TBIPA	This field is used to program the PHY address of the ten-bit interface’s MII management bus. To access the TBI register the user must write the TBIPA value to the MIIMADD [PHY Address] register located in the MAC register section. PHY Address 0 is reserved. Refer to <a href="#">Section 15.5.3.5.8, “MII Management Address Register (MIIMADD)”</a>

### 15.5.3.2 eTSEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

#### 15.5.3.2.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. [Figure 15-11](#) describes the TCTRL register.

Offset eTSEC1:0x2\_4100; eTSEC3:0x2\_6100

Access: Mixed



Figure 15-11. TCTRL Register Definition

Table 15-15 describes the fields of the TCTRL register.

Table 15-15. TCTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	IPCSSEN	IP header checksum generation enable. When set, the eTSEC offloads IPv4 header checksum generation. See <a href="#">Section 15.6.4.2, “Transmit Path Off-Load,”</a> on page 15-158. 0 IP header checksum generation is disabled even if enabled in a transmit frame control block. 1 IP header checksum generation is performed for IPv4 headers as determined by the settings in the current transmit frame control block.

Table 15-15. TCTRL Field Descriptions (continued)

Bits	Name	Description
18	TUCSEN	TCP/UDP header checksum generation enable. When set, the eTSEC offloads TCP or UDP header checksum generation. See <a href="#">Section 15.6.4.2, “Transmit Path Off-Load,”</a> on page 15-158. 0 TCP or UDP header checksum generation is disabled even if enabled in a transmit frame control block. 1 TCP or UDP header checksum generation is performed as determined by the settings in the current transmit frame control block.
19	VLINS	VLAN (IEEE Std. 802.1Q) tag insertion enable. Applicable only for transmission via the Ethernet MAC. 0 Do not insert a VLAN tag into the frame. 1 Insert a VLAN tag into the frame. If the frame FCB has a valid VLAN field, use the FCB to source the VLAN control word, otherwise take the default VLAN control word from register DFVLAN.
20	THDF	Transmit half-duplex flow control under software control for 10-/100-Mbps half-duplex media. This bit is not self-resetting. 0 Disable back pressure 1 Back pressure is applied to media by raising carrier
21–26	—	Reserved
27	RFC_PAUSE	Receive flow control pause frame (written by the eTSEC). This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete. 1 Flow control pause frame received.
28	TFC_PAUSE	Transmit flow control pause frame. Set this bit to transmit a PAUSE frame. If this bit is set, the MAC stops transmission of data frames after the current transmission completes. Next, the GTSC interrupt in the IEVENT register is asserted. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame with the duration value obtained from the PTV register. The TXC interrupt occurs after sending the control pause frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. Note that if the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, the MAC may still transmit a MAC control PAUSE frame. 0 MAC continues. 1 Pause frame is transmitted.

**Table 15-15. TCTRL Field Descriptions (continued)**

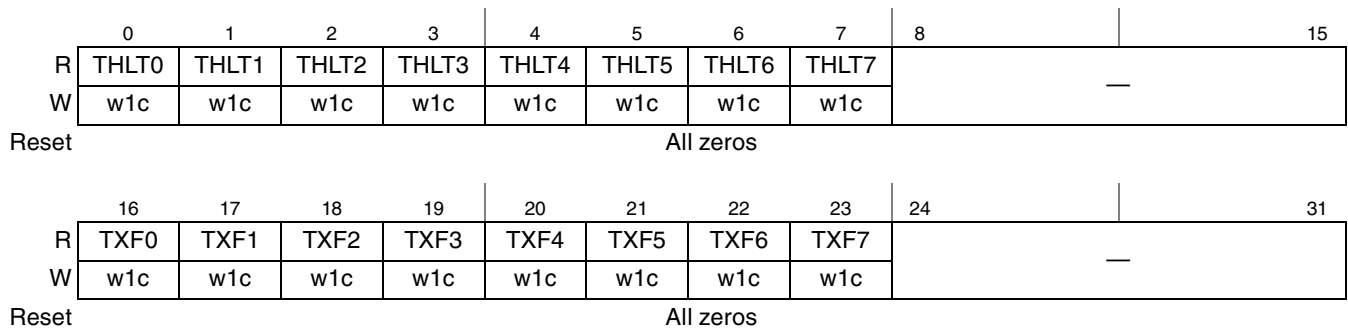
Bits	Name	Description
29–30	TXSCHED	<p>Transmit ring scheduling algorithm. This field determines which scheme the transmit scheduler uses to arbitrate between the enabled TxBD rings. The scheme chosen also controls how the DMACTRL and TQUEUE bits are interpreted. Ring polling is supported only by mode 00; the other modes require software to restart rings with the TSTAT register. TCP/IP offload can be enabled with any scheduling mode.</p> <p>00 Single polled ring mode. TxBD ring 0 is the only ring serviced, even if other rings are enabled and ready. In this scheduler mode, the DMACTRL[WOP] and DMACTRL[TOD] bits control polling and retry behavior. This mode supports ring polling, and allows fetching of a non-ready TxBD to be retried twice.</p> <p>01 Priority scheduling mode. All enabled TxBD rings are serviced in ascending ring index order. Once a non-ready TxBD has been fetched from the lowest-numbered ring, the eTSEC attempts to fetch TxBDs from the next enabled ring having a higher index, until transmission stops for lack of data. TSTAT records whenever a TxBD ring is exhausted.</p> <p>10 Modified weighted round-robin scheduling mode. Each TxBD ring is polled in sequence for frames that are ready for transmission. If a non-ready TxBD is fetched from a ring, that ring is removed from the scheduling pool until software re-enables it. Ready frames are repeatedly transmitted from a chosen ring until its transmission quota is exhausted. The transmission quota for TxBD ring <math>n</math> is set to <math>WT_n \times 64</math> bytes, where <math>WT_n</math> is a weight from the TR03WT/TR47WT registers. If a ring transmits more data than its quota allows, the excess is deducted from its quota on the next transmission opportunity, thereby preventing large frames from monopolizing the eTSEC bandwidth.</p> <p>11 Reserved</p>
31	—	Reserved

**15.5.3.2.2 Transmit Status Register (TSTAT)**

This register is read/write-one-to-clear and is written by the eTSEC to convey DMA status information for each TxBD ring. The halt bit only has meaning for enabled rings. After processing transmit-related interrupts, software should use TSTAT to restart transmission from rings that may have been affected by the interrupt condition. In particular, an error condition that prevents eTSEC from continuing transmission will halt DMA from all rings, including the ring that gave rise to the error. [Figure 15-12](#) describes the TSTAT register.

Offset eTSEC1:0x2\_4104; eTSEC3:0x2\_6104

Access: w1c



**Figure 15-12. TSTAT Register Definition**

Table 15-16 describes the fields of the TSTAT register.

**Table 15-16. TSTAT Field Descriptions**

Bits	Name	Description
0	THLT0	<p>Transmit halt of ring 0. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN0], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include:            Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
1	THLT1	<p>Transmit halt of ring 1. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN1], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include:            Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
2	THLT2	<p>Transmit halt of ring 2. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN2], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include:            Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
3	THLT3	<p>Transmit halt of ring 3. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN3], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include: Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
4	THLT4	<p>Transmit halt of ring 4. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN4], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include: Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
5	THLT5	<p>Transmit halt of ring 5. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN5], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include: Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
6	THLT6	<p>Transmit halt of ring 6. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN6], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include: Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
7	THLT7	<p>Transmit halt of ring 7. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN7], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set.</p> <p>Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again.</p> <p>Repeatable error conditions which cause halt include: Bus error:</p> <ul style="list-style-type: none"> <li>• Invalid BD or data address</li> <li>• Uncorrectable error on BD or data read</li> </ul> <p>TxBD programming errors:</p> <ul style="list-style-type: none"> <li>• Ready=1 and length=0</li> </ul>
8–15	—	Reserved
16	TXF0	Transmit frame event occurred on ring 0. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
17	TXF1	Transmit frame event occurred on ring 1. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
18	TXF2	Transmit frame event occurred on ring 2. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
19	TXF3	Transmit frame event occurred on ring 3. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
20	TXF4	Transmit frame event occurred on ring 4. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
21	TXF5	Transmit frame event occurred on ring 5. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
22	TXF6	Transmit frame event occurred on ring 6. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
23	TXF7	Transmit frame event occurred on ring 7. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
24–31	—	Reserved

### 15.5.3.2.3 Default VLAN Control Word Register (DFVLAN)

This register defines the default value for the VLAN Ethertype and control word when VLAN tags are automatically inserted by the eTSEC, and no per-frame VLAN data is supplied by software. On receive, this register defines a customizable VLAN Ethertype for automatic deletion. Note that an Ethertype of 0x8808 (Control Word) is not permitted as a custom VLAN tag. Frames with an Ethertype of 0x8808 will be dropped by the receiver. In the case of frames containing stacked VLAN tags, this register defines the tag associated with the outer or metropolitan area VLAN. Figure 15-13 describes the DFVLAN register.

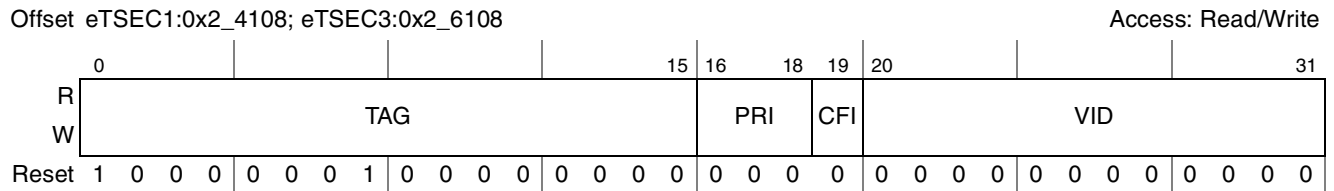


Figure 15-13. DFVLAN Register Definition

Table 15-17 describes the fields of the DFVLAN register.

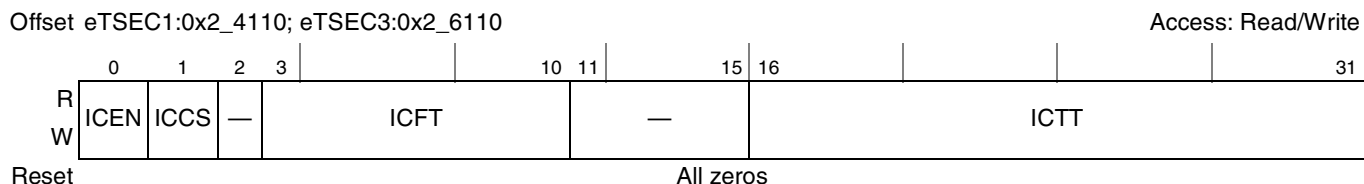
Table 15-17. DFVLAN Field Descriptions

Bits	Name	Description
0–15	TAG	This is the default Ethertype used to tag VLAN frames. On transmit, this tag is inserted ahead of the VLAN control word; TAG should be set to 0x8100 for IEEE 802.1Q VLAN. On receive, an Ethertype matching TAG or an Ethertype of 0x8100 marks a VLAN-tagged frame. Note that if using DFVLAN to set a custom ethertype (that is, using a value other than 0x8100), packets received with a custom tag are not counted by any of the RMON counters. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF.
16–18	PRI	This is the default value used for the IEEE Std. 802.1p frame priority.
19	CFI	This is the default value used for the IEEE Std. 802.1Q canonical format indicator.
20–31	VID	This is the default value used for the virtual-LAN identifier in VLAN-tagged frames. A value of zero is defined as the null VLAN, however field PRI may be still set independently.



### 15.5.3.2.4 Transmit Interrupt Coalescing Register (TXIC)

The TXIC register enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Figure 15-14 describes the definition for the TXIC register.



**Figure 15-14. TXIC Register Definition**

Table 15-18 describes the fields of the TXIC register.

**Table 15-18. TXIC Field Descriptions**

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by TXIC[ICFT]) or when the threshold timer expires (determined by TXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Tx interface clocks (TSEC <sub>n</sub> _GTX_CLK). 1 The coalescing timer advances count every 64 system clocks. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines how many frames are transmitted before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero to avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines the maximum amount of time after transmitting a frame before raising an interrupt. If frames have been transmitted but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon transmission of the first frame having its TxBD[1] bit set. The threshold value is represented in units of 64 clock periods as specified by the timer clock source (TXIC[ICCS]). The value of ICTT must be greater than zero to avoid unpredictable behavior.

### 15.5.3.2.5 Transmit Queue Control Register (TQUEUE)

The TQUEUE register, shown in Figure 15-15, selectively enables each of the TxBD rings 0–7. By default, TxBD ring 0 is enabled.

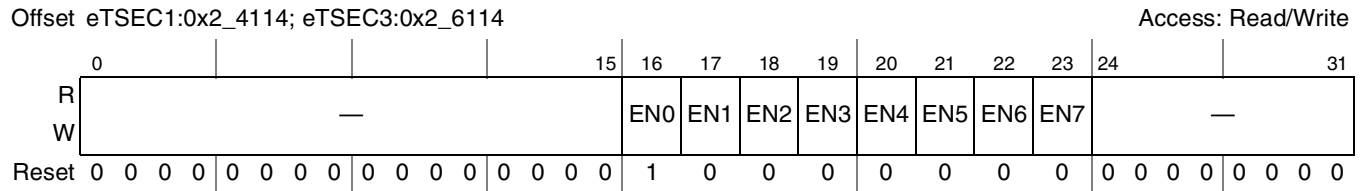


Figure 15-15. TQUEUE Register Definition

Table 15-19 describes the TQUEUE register.

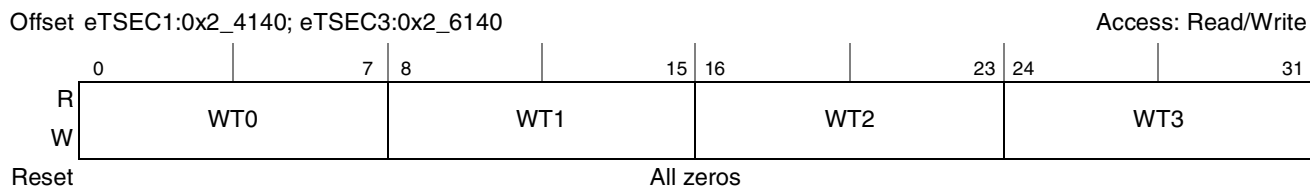
Table 15-19. TQUEUE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	EN0	Transmit queue 0 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
17	EN1	Transmit queue 1 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
18	EN2	Transmit queue 2 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
19	EN3	Transmit queue 3 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
20	EN4	Transmit queue 4 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
21	EN5	Transmit queue 5 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
22	EN6	Transmit queue 6 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
23	EN7	Transmit queue 7 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
24–31	—	Reserved

### 15.5.3.2.6 TxBD Ring 0–3 Weighting Register (TR03WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHEM] = 10), this register determines the weighting applied to each transmit queue for queues 0 to 3. For priority-based scheduling,

TR03WT has no effect. A description of how queue weights affect eTSEC's round-robin algorithm appears in [Section 15.6.5.2.2, "Modified Weighted Round-Robin Queuing \(MWRR\)."](#) Figure 15-16 describes the TR03WT register.



**Figure 15-16. TR03WT Register Definition**

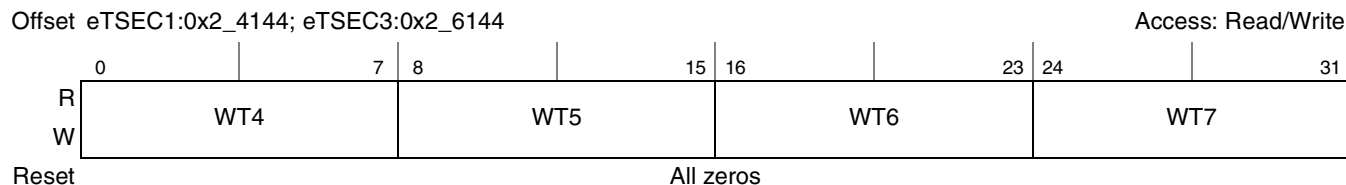
Table 15-20 describes the fields of the TR03WT register.

**Table 15-20. TR03WT Field Descriptions**

Bits	Name	Description
0–7	WT0	Weighting value for TxBD ring 0 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of $WT0 \times 64$ bytes of data are scheduled for transmission from TxBD ring 0. Clearing this field prevents transmission.
8–15	WT1	Weighting value for TxBD ring 1 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of $WT1 \times 64$ bytes of data are scheduled for transmission from TxBD ring 1. Clearing this field prevents transmission.
16–23	WT2	Weighting value for TxBD ring 2 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of $WT2 \times 64$ bytes of data are scheduled for transmission from TxBD ring 2. Clearing this field prevents transmission.
24–31	WT3	Weighting value for TxBD ring 3 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of $WT3 \times 64$ bytes of data are scheduled for transmission from TxBD ring 3. Clearing this field prevents transmission.

### 15.5.3.2.7 TxBD Ring 4–7 Weighting Register (TR47WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHED] = 10), this register determines the weighting applied to each enabled transmit queue for queues 4 to 7. For priority-based scheduling, TR47WT has no effect. A description of how queue weights affect eTSEC's modified weighted round-robin algorithm appears in [Section 15.6.5.2.2, "Modified Weighted Round-Robin Queuing \(MWRR\)."](#) Figure 15-17 describes the definition for the TR47WT register.



**Figure 15-17. TR47WT Register Definition**

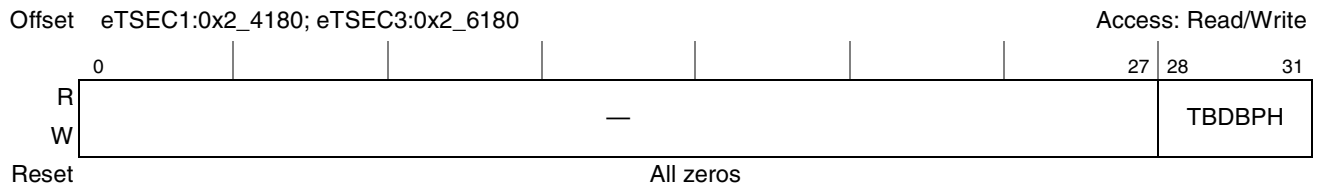
Table 15-21 describes the fields of the TR47WT register.

**Table 15-21. TR47WT Field Descriptions**

Bits	Name	Description
0–7	WT4	Weighting value for TxBd ring 4 when TCTRL[TXSCHEd] = 10. On each round of the Tx scheduler, a minimum of WT4 × 64 bytes of data are scheduled for transmission from TxBd ring 4. Clearing this field prevents transmission.
8–15	WT5	Weighting value for TxBd ring 5 when TCTRL[TXSCHEd] = 10. On each round of the Tx scheduler, a minimum of WT5 × 64 bytes of data are scheduled for transmission from TxBd ring 5. Clearing this field prevents transmission.
16–23	WT6	Weighting value for TxBd ring 6 when TCTRL[TXSCHEd] = 10. On each round of the Tx scheduler, a minimum of WT6 × 64 bytes of data are scheduled for transmission from TxBd ring 6. Clearing this field prevents transmission.
24–31	WT7	Weighting value for TxBd ring 7 when TCTRL[TXSCHEd] = 10. On each round of the Tx scheduler, a minimum of WT7 × 64 bytes of data are scheduled for transmission from TxBd ring 7. Clearing this field prevents transmission.

### 15.5.3.2.8 Transmit Data Buffer Pointer High Register (TBDBPH)

The TBDBPH register is written by the user with the most significant address bits common to all TxBd buffer addresses, TxBd[Data Buffer Pointer]. As a consequence, all Tx buffers must be placed in a 4 gigabyte segment of memory whose base address is prefixed by the bits in TBDBPH. The TxBd ring itself can reside in a different memory region (based at TBASEH). Figure 15-18 describes the definition for the TBDBPH register.



**Figure 15-18. TBDBPH Register Definition**

Table 15-24 describes the fields of the TBDBPH register.

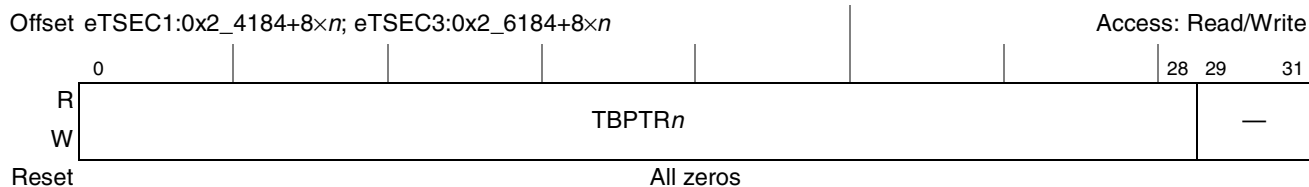
**Table 15-22. TBDBPH Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	TBDBPH	Most significant bits common to all data buffer addresses contained in TxBds. The user must initialize TBDBPH before enabling the eTSEC transmit function.

### 15.5.3.2.9 Transmit Buffer Descriptor Pointers 0–7 (TBPTR0–TBPTR7)

TBPTR0–TBPTR7 each contains the low-order 32 bits of the next transmit buffer descriptor address for their respective TxBd ring. Figure 15-19 describes the TBPTR registers. These registers takes on the value of their ring’s associated TBASE when the TBASE register is written by software. Software must not write TBPTR0–TBPTR7 while eTSEC is actively transmitting frames. However, TBPTR0– TBPTR7 can be modified when the transmitter is disabled or when no Tx buffer is in use (after a GRACEFUL STOP

TRANSMIT command is issued and the frame completes its transmission) in order to change the next TxBD eTSEC transmits.



**Figure 15-19. TBPTR0–TBPTR7 Register Definition**

Table 15-23 describes the fields of the TBPTR<sub>*n*</sub> register.

**Table 15-23. TBPTR<sub>*n*</sub> Field Descriptions**

Bits	Name	Description
0–28	TBPTR <sub><i>n</i></sub>	Current TxBD pointer for TxBD ring <i>n</i> . Points to the current BD being processed or to the next BD the transmitter uses when it is idling. When the end of the TxBD ring is reached, eTSEC initializes TBPTR <sub><i>n</i></sub> to the value in the corresponding TBASE <sub><i>n</i></sub> . The TBPTR register is internally written by the eTSEC's DMA controller during transmission. The pointer increments by eight (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the three least significant bits of this register are read-only and zero. After an error condition, the eTSEC returns TBPTR <sub><i>n</i></sub> to point to the first BD of the frame partially transmitted.
29–31	—	Reserved

### 15.5.3.2.10 Transmit Descriptor Base Address High Register (TBASEH)

The TBASEH register is written by the user with the most significant address bits common to all TxBD addresses, including TBASE0–TBASE7 and TBPTR0–TBPTR7. As a consequence, all TxBD rings must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in TBASEH. Data buffers are located in a potentially different region, based at TBDBPH. Figure 15-20 describes the TBASEH register.



**Figure 15-20. TBASEH Register Definition**

Table 15-24 describes the fields of the TBASEH register.

**Table 15-24. TBASEH Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	TBASEH	Most significant bits common to all TxBD addresses—except data buffer pointers. The user must initialize TBASEH before enabling the eTSEC transmit function.

### 15.5.3.2.11 Transmit Descriptor Base Address Registers (TBASE0–TBASE7)

The TBASE $n$  registers are written by the user with the base address of each TxBD ring  $n$ . Each such value must be divisible by eight, since the three least significant bits always write as 000. Figure 15-21 describes the definition for the TBASE $n$  registers.

Offset eTSEC1:0x2\_4204+8 $\times$  $n$ ; eTSEC3:0x2\_6204+8 $\times$  $n$

Access: Read/Write

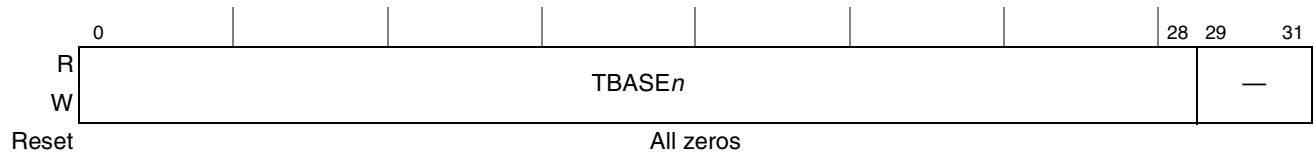


Figure 15-21. TBASE Register Definition

Table 15-25 describes the fields of the TBASE $n$  registers.

Table 15-25. TBASE0–TBASE7 Field Descriptions

Bits	Name	Description
0–28	TBASE $n$	Transmit base for ring $n$ . TBASE defines the starting location in the memory map for the eTSEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the eTSEC transmit function on the associated ring.
29–31	—	Reserved

### 15.5.3.3 eTSEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

#### 15.5.3.3.1 Receive Control Register (RCTRL)

The RCTRL register is programmed by the user and controls the operational mode of the receiver. It must be written only after a system reset (at initialization) or after a graceful receive stop has completed. Figure 15-22 describes the RCTRL register.

Offset eTSEC1:0x2\_4300; eTSEC3:0x2\_6300

Access: Read/Write

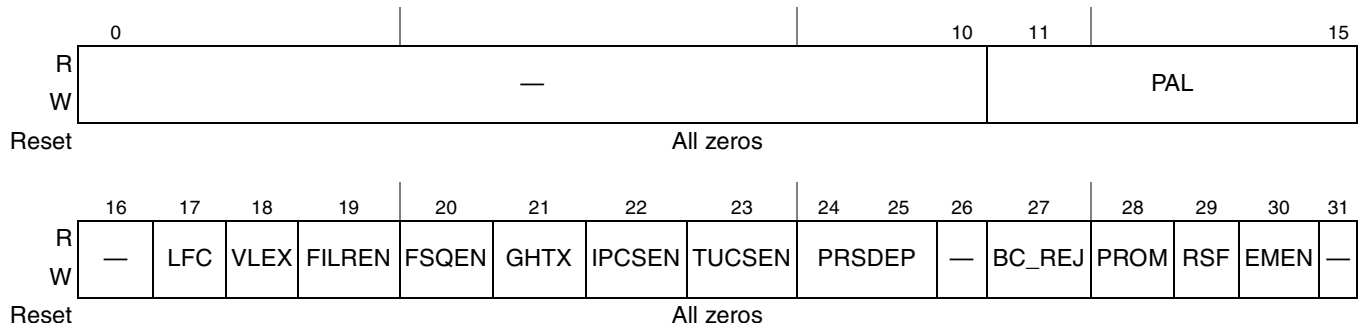


Figure 15-22. RCTRL Register Definition

Table 15-26 describes the fields of the RCTRL register.

**Table 15-26. RCTRL Field Descriptions**

Bits	Name	Description
0–10	—	Reserved
11–15	PAL	Packet alignment padding length. If not zero, PAL (1–31) bytes of zero padding are inserted before the start of each received frame, but following the RxFCB if TOE is enabled. For Ethernet where optional preamble extraction is enabled, the padding appears before the preamble, otherwise the padding precedes the layer 2 header. The value of PAL can be set so that the start of the IP header in the receive data buffer is aligned to a 32-bit boundary. Normally, setting PAL = 2 will provide minimal padding to ensure such alignment of the IP header.
17	LFC	Lossless flow control. When set, the eTSEC will determine the number of free BDs (via RQPARM $n$ [LEN] and RBTPT $Rn$ ) in each active ring. Should the free BD count in an active ring drop below its setting for RQPARM $n$ [FBTHR], the eTSEC will assert link layer flow control. For full-duplex ethernet connections, the eTSEC will emit a pause frame as if TCTRL[TFC_PAUSE] was set. For FIFO packet interface connections, the RFC signal will be asserted. 0 Disabled. This is the default 1 Enabled, calculate the free BDs in each active ring and assert link layer flow control if required.
18	VLEX	Enable automatic VLAN tag extraction and deletion from Ethernet frames. 0 Do not delete VLAN tags from received Ethernet frames. 1 If a VLAN tag is seen after the Ethernet source address, and PRSDEP is non-zero, delete the VLAN tag and return the VLAN control word in the frame control block returned with this frame. Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.)
19	FILREN	Filer enable. When set, the receive frame filer is enabled. This will file accepted frames to a particular RxBD ring according to rules defined in the filer table. In this case, PRSDEP must not be cleared. 0 Do not search the receive queue filer table for received frames. All received frames are sent to RxBD ring 0 by default. 1 Search the receive queue filer table for received frames, and let the filer determine the index of the RxBD ring for each frame. Note that if PRSDEP is cleared, FILREN must be cleared as well.
20	FSQEN	Enable single-queue mode for the receive frame filer. This bit is ignored unless FILREN is also set. 0 The filer chooses the RxBD ring using the least significant bits of the virtual queue ID as a ring index. 1 The filer always attempts to file received frames to ring 0, regardless of virtual queue ID. This mode is intended for operating the filer as a packet classification engine.
21	GHTX	Group address hash table extend. By default, the group address hash table is 256 entries (as defined by registers GADDR0–GADDR7); registers IGADDR0–IGADDR7 are then used to define the individual address hash table. When this bit is set, the hash table is extended to a total of 512 entries (IGADDR0–IGADDR7 are then the first 256 entries of the extended 512-entry group address hash table). 0 Both the individual and group hash functions are the 8 MSBs of the CRC-32 of the Ethernet destination address. 1 The group hash function is the 9 MSBs of the CRC-32 of the Ethernet destination address. The individual address hash function is unavailable.
22	IPCSEN	IP Checksum verification enable. See <a href="#">Section 15.6.4.3, “Receive Path Off-Load.”</a> 0 IPv4 header checksums are not verified by the eTSEC—even if layer 3 parsing is enabled. 1 Perform IPv4 header checksum verification if PRSDEP > 01.
23	TUCSEN	TCP or UDP Checksum verification enable. See <a href="#">Section 15.6.4.3, “Receive Path Off-Load.”</a> 0 TCP or UDP checksums are not verified by the eTSEC—even if layer 4 parsing is enabled. 1 Perform TCP or UDP checksum verification if PRSDEP = 11.

Table 15-26. RCTRL Field Descriptions (continued)

Bits	Name	Description
24–25	PRSDEP	<p>Parser control. The level of parser layer recognition is determined as follows:</p> <p>00 Parser disabled. Receive frame filter must also be disabled by clearing RCTRL[FILREN]. This should be the setting for raw (non-IP) packets received over FIFO interface</p> <p>01 Only L2 (ethernet) protocols are recognized. For packets received over FIFO interface, this parse level is unavailable.</p> <p>10 L2 and L3 (IP) protocols are recognized. This is the minimum parse level for IP packets received over FIFO interface.</p> <p>11 L2, L3 and L4 (TCP/UDP) protocols are recognized.</p> <p>If this field is non-zero, a TOE frame control block is prepended to the received frame, and the first RxBD will point to the FCB.</p> <p>Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.) Also, if PRSDEP is cleared, FILREN must also be cleared.</p>
26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD.
28	PROM	Promiscuous mode. All Ethernet frames, regardless of destination address, are accepted.
29	RSF	<p>Receive short frame mode. When set, enables the reception of frames shorter than 64 bytes. For packets received over the FIFO packet interface, this bit has no effect (packets shorter than 64 bytes are always accepted).</p> <p>0 Ethernet frames less than 64B in length are silently dropped.</p> <p>1 Frames more than 16B and less than 64B in length are accepted upon a DA match.</p> <p>Note that frames less than or equal to 16B in length are always silently dropped.</p>
30	EMEN	Exact match MAC address enable. If this bit is set, the MAC01ADDR1–MAC15ADDR1 and MAC01ADDR2–MAC15ADDR2 registers are recognized as containing MAC addresses aliasing the MAC's station address. Setting this bit therefore allows eTSEC to receive Ethernet frames having a destination address matching one of these 15 addresses.
31	—	Reserved

### 15.5.3.3.2 Receive Status Register (RSTAT)

The eTSEC writes to this register under the following conditions:

- A frame interrupt event occurred on one or more RxBD rings
- The receiver runs out of descriptors due to a busy condition on a RxBD ring
- The receiver was halted because an error condition was encountered while receiving a frame



Writing 1 to any bit of this register clears it. Software should clear the QHLT bit to take eTSEC's receiver function out of halt state for the associated queue. [Figure 15-23](#) describes the definition for the RSTAT register.

Offset eTSEC1:0x2\_4304; eTSEC3:0x2\_6304

Access: w1c

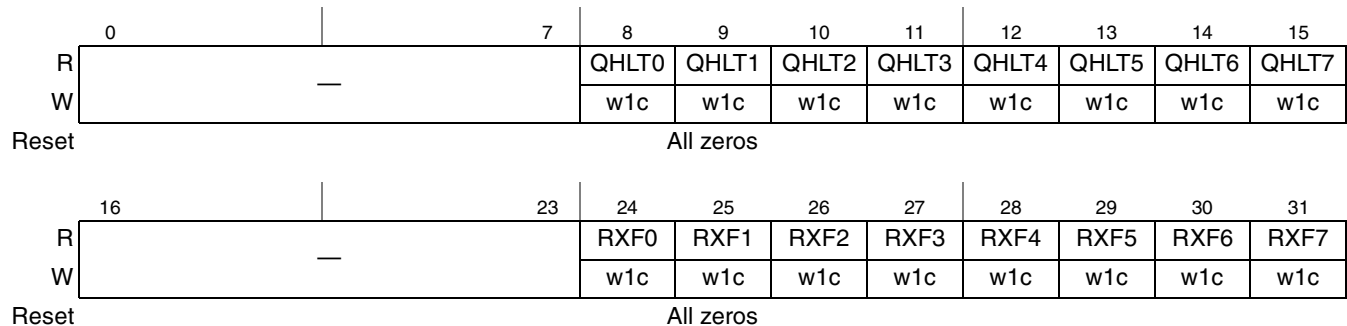


Figure 15-23. RSTAT Register Definition

[Table 15-27](#) describes the fields of the RSTAT register.

Table 15-27. RSTAT Field Descriptions

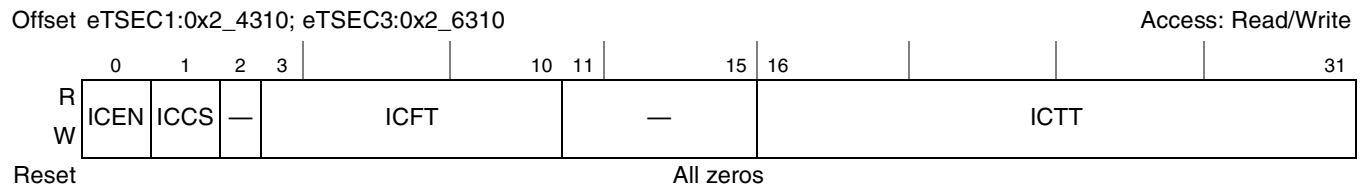
Bits	Name	Description
0–7	—	Reserved
8	QHLT0	RxBD queue 0 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT0 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
9	QHLT1	RxBD queue 1 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT1 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
10	QHLT2	RxBD queue 2 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT2 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
11	QHLT3	RxBD queue 3 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT3 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
12	QHLT4	RxBD queue 4 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT4 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.

**Table 15-27. RSTAT Field Descriptions (continued)**

Bits	Name	Description
13	QHLT5	RxBD queue 5 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT5 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
14	QHLT6	RxBD queue 6 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT6 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
15	QHLT7	RxBD queue 7 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT7 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
16–23	—	Reserved
24	RXF0	Receive frame event occurred on ring 0. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
25	RXF1	Receive frame event occurred on ring 1. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
26	RXF2	Receive frame event occurred on ring 2. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
27	RXF3	Receive frame event occurred on ring 3. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
28	RXF4	Receive frame event occurred on ring 4. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
29	RXF5	Receive frame event occurred on ring 5. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
30	RXF6	Receive frame event occurred on ring 6. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
31	RXF7	Receive frame event occurred on ring 7. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.

**15.5.3.3 Receive Interrupt Coalescing Register (RXIC)**

The RXIC register enables and configures the operational parameters for interrupt coalescing associated with received frames. [Figure 15-24](#) describes the RXIC register.



**Figure 15-24. RXIC Register Definition**

Table 15-28 describes the fields of the RXIC register.

**Table 15-28. RXIC Field Descriptions**

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC receive frame interrupt is enabled (IMASK[RXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by RXIC[ICFT]) or when the threshold timer expires (determined by RXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Rx interface clocks (TSECn_GTX_CLK). 1 The coalescing timer advances count every 64 system clocks. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines how many frames are received before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines the maximum amount of time after receiving a frame before raising an interrupt. If frames have been received but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon receiving the first frame having its RxBd[I] bit set. The threshold value is represented in units equal to 64 periods of the clock specified by RXIC[ICCS]. ICTT must be greater than zero to avoid unpredictable behavior.

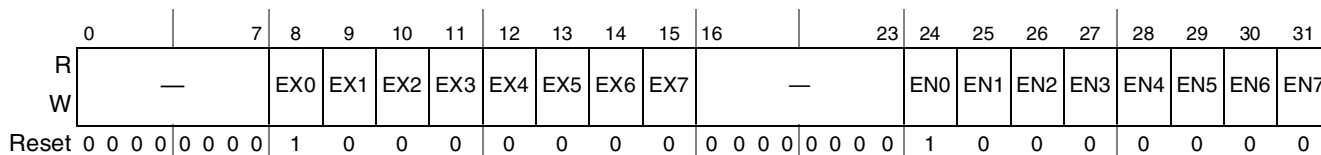
#### 15.5.3.3.4 Receive Queue Control Register (RQUEUE)

The RQUEUE register enables each of the RxBd rings 0–7. By default, RxBd ring 0 is enabled.

Figure 15-25 describes the definition for the RQUEUE register.

Offset eTSEC1:0x2\_4314; eTSEC3:0x2\_6314

Access: Read/Write



**Figure 15-25. RQUEUE Register Definition**

Table 15-29 describes the RQUEUE register.

**Table 15-29. RQUEUE Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8	EX0	Receive queue 0 extract enable. 0 Data transferred by DMA to this RxBd ring is not extracted to cache. 1 Data transferred by DMA to this RxBd ring undergoes extraction according to ATTR register.

Table 15-29. RQUEUE Field Descriptions (continued)

Bits	Name	Description
9	EX1	Receive queue 1 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
10	EX2	Receive queue 2 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
11	EX3	Receive queue 3 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
12	EX4	Receive queue 4 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
13	EX5	Receive queue 5 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
14	EX6	Receive queue 6 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
15	EX7	Receive queue 7 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
16–23	—	Reserved
24	EN0	Receive queue 0 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
25	EN1	Receive queue 1 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
26	EN2	Receive queue 2 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
27	EN3	Receive queue 3 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
28	EN4	Receive queue 4 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
29	EN5	Receive queue 5 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.

Table 15-29. RQUEUE Field Descriptions (continued)

Bits	Name	Description
30	EN6	Receive queue 6 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
31	EN7	Receive queue 7 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.

### 15.5.3.3.5 Receive Bit Field Extract Control Register (RBIFX)

The RBIFX register provides a set of four 6-bit offsets for locating up to four octets in a received frame and passing them to the receive queue filer as the user-defined ARB property. Through RBIFX a custom ARB filer property can be constructed from arbitrary bytes, which allows frame filing on the basis of bitfields not ordinarily provided to the filer, such as bits from the Ethernet preamble or TCP flags. The value of property ARB is the concatenation of {B0, B1, B2, B3} to 32-bits, where B0–B3 are the bytes as defined by RBIFX.

Figure 15-26 describes the definition for the RBIFX register. Note: when the eTSEC is configured to receive frame via the FIFO packet interface, a value of BnCTL = 01 is not supported unless RCTRL[PRFSFM]=1. In addition, the byte extraction level cannot exceed the parser depth: a value of BnCTL=10 requires RCTRL[PRSDEP]=1x and a value of BnCTL=11 requires RCTRL[PRSDEP]=11.

For values of BnCTL=10 or BCTL=11, the controller will extract the defined bytes even if it does not recognize the L3 or L4 header, respectively.

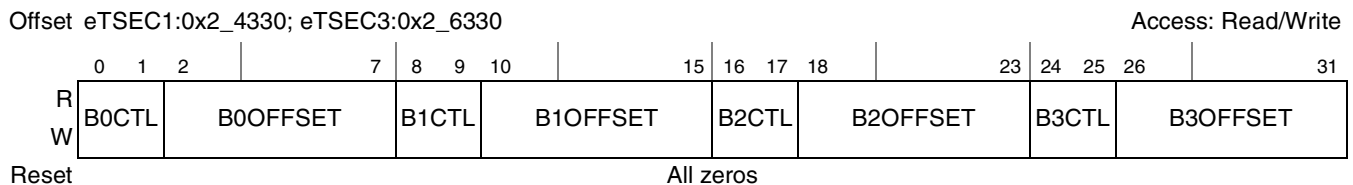


Figure 15-26. RBIFX Register Definition

Table 15-30 describes the RBIFX register.

Table 15-30. RBIFX Field Descriptions

Bits	Name	Description
0–1	B0CTL	Location of byte 0 of property ARB. 00 Byte 0 is not extracted, and appears as zero in property ARB. 01 Byte 0 is located in the received frame at offset (B0OFFSET – 8) bytes from the first byte of the Ethernet DA. A negative effective offset points to bytes of the standard Ethernet preamble. 10 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 3 header.
2–7	B0OFFSET	Offset relative to the header defined by B0CTL that locates byte 0 of property ARB. An effective offset of zero points to the first byte of the specified header.

Table 15-30. RBIFX Field Descriptions (continued)

Bits	Name	Description
8–9	B1CTL	Location of byte 1 of property ARB. 00 Byte 1 is not extracted, and appears as zero in property ARB. 01 Byte 1 is located in the received frame at offset (B1OFFSET – 8) bytes from the first byte of the Ethernet DA. A negative effective offset points to bytes of the standard Ethernet preamble. 10 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 3 header.
10–15	B1OFFSET	Offset relative to the header defined by B1CTL that locates byte 1 of property ARB. An effective offset of zero points to the first byte of the specified header.
16–17	B2CTL	Location of byte 2 of property ARB. 00 Byte 2 is not extracted, and appears as zero in property ARB. 01 Byte 2 is located in the received frame at offset (B2OFFSET – 8) bytes from the first byte of the Ethernet DA. A negative effective offset points to bytes of the standard Ethernet preamble. 10 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 3 header.
18–23	B2OFFSET	Offset relative to the header defined by B2CTL that locates byte 2 of property ARB. An effective offset of zero points to the first byte of the specified header.
24–25	B3CTL	Location of byte 3 of property ARB. 00 Byte 3 is not extracted, and appears as zero in property ARB. 01 Byte 3 is located in the received frame at offset (B3OFFSET – 8) bytes from the first byte of the Ethernet DA. A negative effective offset points to bytes of the standard Ethernet preamble. 10 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 3 header.
26–31	B3OFFSET	Offset relative to the header defined by B3CTL that locates byte 3 of property ARB. An effective offset of zero points to the first byte of the specified header.

15.5.3.3.6 Receive Queue Filer Table Address Register (RQFAR)

RQFAR, shown in Figure 15-27, contains the index of the current, indirectly accessible entry of the received queue filer table. Each table entry occupies a pair of 32-bit words, denoted RQCTRL and RQPROP. To access the RQCTRL and RQPROP words of entry *n*, write *n* to RQFAR. Then read or write the indexed RQCTRL and RQPROP words by reading or writing the RQFCR and RQFPR registers, respectively.

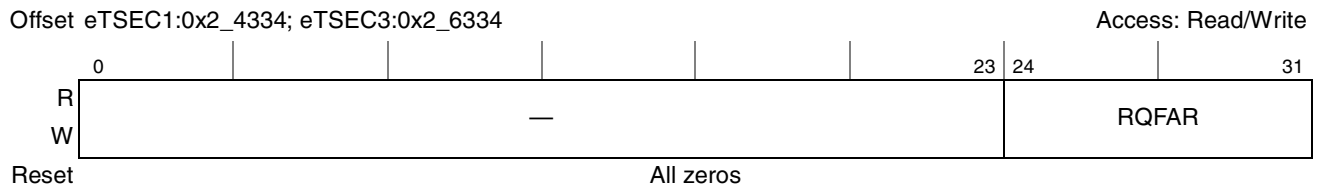


Figure 15-27. Receive Queue Filer Table Address Register Definition

Table 15-31 describes the fields of the RQFAR register.

**Table 15-31. RQFAR Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24–31	RQFAR	Current index of receive queue filer table, which spans a total of 256 entries.

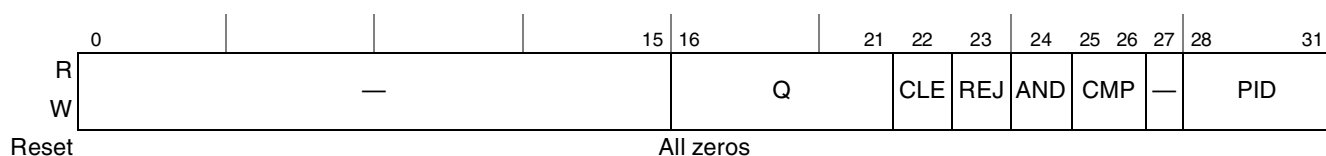
### 15.5.3.3.7 Receive Queue Filer Table Control Register (RQFCR)

RQFCR is accessed to read or write the RQCTRL words in entries of the receive queue filer table. The table entries are described in greater detail in Section 15.6.5.1, “Receive Queue Filer.” The word accessed via RQFCR is defined by the current value of RQFAR.

Figure 15-28 describes the definition for the RQFCR register.

Offset eTSEC1:0x2\_4338; eTSEC3:0x2\_6338

Access: Read/Write



**Figure 15-28. Receive Queue Filer Table Control Register Definition**

Table 15-32 describes the fields of the RQFCR register.

**Table 15-32. RQFCR Field Descriptions**

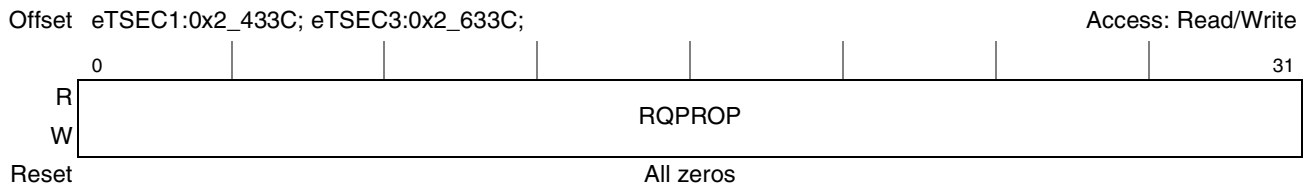
Bit	Name	Description
0–15	—	Reserved, should be written with zero.
16–21	Q	Receive queue index, from 0 to 63, inclusive, written into the Rx frame control block associated with the received frame. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the frame is sent to either RxBD ring 0 (if RCTRL[FSQEN] = 1) or the RxBD ring with index (Q mod 8) and the filing table search is terminated. In the case where RCTRL[FSQEN] = 0, 8 virtual receive queues are overlaid on every RxBD ring, and software needs to consult the RQ field of the Rx frame control block to determine which virtual receive queue was chosen.
22	CLE	Cluster entry/exit (used in combination with AND bit). This bit brackets clusters, marking the start and end entries of a cluster. Clusters cannot be nested. 0 Regular RQCTRL entry. 1 If entry matches and AND = 1, treat subsequent entries as belonging to a nested cluster and enter the cluster; otherwise skip all entries up to and including the next cluster exit. If AND = 0, exit current cluster.
23	REJ	Reject frame. This bit and its specified action are ignored if AND = 1. 0 If entry matches, accept frame and file it to RxBD ring Q. 1 If entry matches, reject frame and discard it, ignoring Q.
24	AND	Match this entry and the next entry as a pair. 0 Match property[PID] against RQPROP, independent of the next entry. 1 Match property[PID] against RQPROP. If matched and CLE = 0, attempt to match next entry, otherwise, skip all entries up to and including the entry with AND = 0. If matched and CLE = 1, enter cluster of entries, otherwise, skip all entries up to and including the entry with CLE = 1 (cluster exit).

**Table 15-32. RQFCR Field Descriptions (continued)**

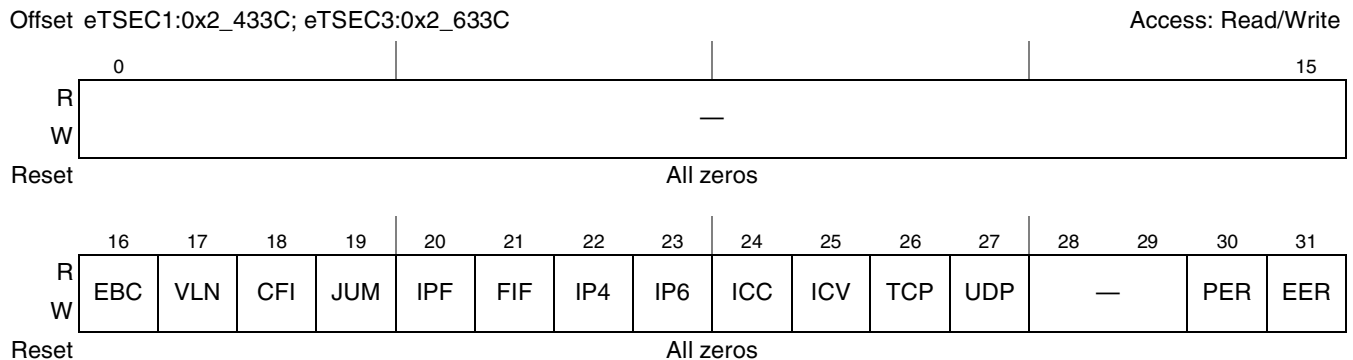
Bit	Name	Description
25–26	CMP	<p>Comparison operation to perform on the RQPROP entry at this index when PID &gt; 0. The property value extracted by the frame parser is masked by the 32-bit <i>mask_register</i> prior to comparison against RQPROP. However, the property value is not permanently altered by the value in <i>mask_register</i>. By default, <i>mask_register</i> is initialized to 0xFFFF_FFFF before each frame is processed.</p> <p>In the case where PID = 0, CMP is interpreted as follows:                      00/01 Filer <i>mask_register</i> is set to all 32 bits of RQPROP, and this entry always <i>matches</i>.                      10/11 Filer <i>mask_register</i> is set to all 32 bits of RQPROP, and this entry always <i>fails to match</i>.</p> <p>In the case where PID &gt; 0, CMP is interpreted as follows (&amp; is bit-wise AND operator):                      00 <i>property</i>[PID] &amp; <i>mask_register</i> = RQPROP                      01 <i>property</i>[PID] &amp; <i>mask_register</i> &gt;= RQPROP                      10 <i>property</i>[PID] &amp; <i>mask_register</i> != RQPROP                      11 <i>property</i>[PID] &amp; <i>mask_register</i> &lt; RQPROP</p>
27	—	Reserved, should be written with zero.
28–31	PID	Property identifier. The value in the RQPROP entry at this index is interpreted according to PID (see <a href="#">Table 15-33</a> ).

**15.5.3.3.8 Receive Queue Filer Table Property Register (RQFPR)**

RQFPR (see [Figure 15-29](#)) is accessed to read or write the RQPROP words in entries of the receive queue filer table. The table entries are described in greater detail in [Section 15.6.5.1, “Receive Queue Filer.”](#) The word accessed via RQFPR is defined by the current value of RQFAR. [Figure 15-29](#) and [Figure 15-30](#) describe the fields of the RQFPR register according to property ID.



**Figure 15-29. Receive Queue Filer Table Property IDs 0, 2–15 Register Definition**



**Figure 15-30. Receive Queue Filer Table Property ID1 Register Definition**



Table 15-33 describes the fields of the RQFPR register.

**Table 15-33. RQFPR Field Descriptions**

PID <sup>1</sup>	Bit	Name	Description
0000	0–31	MASK	Mask bits to be written to Filer <i>mask_register</i> for masking of property values. The rule match/fail status for this PID is determined by RQCTRL[ <i>CMP</i> ]. Since <i>mask_register</i> is bit-wise ANDed with properties, every bit of MASK that is cleared also results in the corresponding property bit being cleared in comparisons. Therefore setting MASK to 0xFFFF_FFFF ensures that all property bits participate in rule matches.
0001	0–15	—	Reserved
	16	EBC	Set if the destination Ethernet address is to the broadcast address.
	17	VLN	Set if a VLAN tag (Ethertype DFVLAN[ <i>TAG</i> ] or 0x8100) was seen in the frame.
	18	CFI	Set to the value of the Canonical Format Indicator in the VLAN control tag if VLAN is set, zero otherwise.
	19	JUM	Set if a jumbo Ethernet frame was parsed.
	20	IPF	Set if a fragmented IPv4 or IPv6 header was encountered. See the descriptions of receive FCB fields IP and PRO in <a href="#">Section 15.6.4.3, “Receive Path Off-Load,”</a> for more information on determining the status of received packets for which IPF is set.
	21	FIF	Set if the packet entered on eTSEC’s FIFO interface.
	22	IP4	Set if an IPv4 header was parsed.
	23	IP6	Set if an IPv6 header was parsed.
	24	ICC	Set if the IPv4 header checksum was checked.
	25	ICV	Set if the IPv4 header checksum was verified correct.
	26	TCP	Set if a TCP header was parsed.
	27	UDP	Set if a UDP header was parsed.
	28–29	—	Reserved.
	0010	0–7	ARB
8–15		User-defined arbitrary bit field property: byte 1 extracted. Defaults to 0x00.	
16–23		User-defined arbitrary bit field property: byte 2 extracted. Defaults to 0x00.	
24–31		User-defined arbitrary bit field property: byte 3 extracted. Defaults to 0x00.	
0011	0–7	—	Reserved, should be written with zero.
	8–31	DAH	Destination MAC address, most significant 24 bits. Defaults to 0x000000.
0100	0–7	—	Reserved, should be written with zero.
	8–31	DAL	Destination MAC address, least significant 24 bits. Defaults to 0x000000.
0101	0–7	—	Reserved, should be written with zero.
	8–31	SAH	Source MAC address, most significant 24 bits. Defaults to 0x000000.

Table 15-33. RQFPR Field Descriptions (continued)

PID <sup>1</sup>	Bit	Name	Description
0110	0–7	—	Reserved, should be written with zero.
	8–31	SAL	Source MAC address, least significant 24 bits. Defaults to 0x000000.
0111	0–15	—	Reserved, should be written with zero.
	16–31	ETY	<p>Ethertype of next layer protocol, i.e., last ethertype if layer 2 headers nest. Defaults to 0xFFFF.</p> <p>Using the filer to match ETY does not work in the case of PPPoE packets, because the PPPoE ethertype in the original packet, 0x8864, is always overwritten with the PPP protocol field. Thus, matches on ETY == 0x8864 always fail.</p> <p>Instead, software should use PID=1 fields IP4 (ETY = 0x0021) and IP6 (ETY = 0x0057) to distinguish PPPoE session packets carrying IPv4 and IPv6 datagrams. Other PPP protocols are encoded in the ETY field, but many of them overlap with real ethertype definitions. Consult IANA and IEEE for possible ambiguities.</p> <p>Packets with a value in the length/type field greater than 1500 and less than 1536 are treated as payload length. If the eTSEC is used in a network where there are packets carrying a type designation between 1500 and 1536 (note there are none currently publicly defined by IANA), then the S/W must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.</p> <p>Note that the eTSEC filer gets multiple packet attributes as a result of parsing the packet. The behavior of the eTSEC is that it will pull the innermost ethertype found in the packet; this means that in many supported protocols, it is impossible to create a filer rule that will match on the outer ethertype. There are four cases that need to be highlighted.</p> <ol style="list-style-type: none"> <li>1. The jumbo ethertype (0x8870)—In this case, the eTSEC assumes that the following header is LLC/SNAP. LLC/SNAP has an associated ethertype, and the ETY field will be populated with that ethertype. This makes it impossible to file on jumbo frames. In this case, one can use arbitrary extracted bytes to pull the outermost Ethertype.</li> <li>2. The PPPoE ethertype described above.</li> <li>3. The VLAN tag ethertype (0x8100)—In this case, one can use the PID1 VLN bit to indicate that the packet had a VLAN tag.</li> <li>4. The MPLS tagged packets. In this case, one can use arbitrary extraction bytes to compare to the actual ethertype if a filer rule is intending to file based on an MPLS label existence.</li> </ol>
1000	0–19	—	Reserved, should be written with zero.
	20–31	VID	VLAN network identifier (as per IEEE Std. 802.1Q). This value defaults to 0x000 if no VLAN tag was found, or the VLAN tag contained only priority information.
1001	0–28	—	Reserved, should be written with zero.
	29–31	PRI	VLAN user priority (as per IEEE Std. 802.1p). This value defaults to 000 (best effort priority) if no VLAN tag was found.
1010	0–23	—	Reserved, should be written with zero.
	24–31	TOS	IPv4 header Type Of Service field or IPv6 Traffic Class field. This value defaults to 0x00 (default RFC 2474 best-effort behavior) if no IP header appeared. Note that for IPv6 the Traffic Class field is extracted using the IP header definition in RFC 2460. IPv6 headers formed using the earlier RFC 1883 have a different format and must be handled with software.
1011	0–23	—	Reserved, should be written with zero.
	24–31	L4P	Layer 4 protocol identifier as per published IANA specification. This is the last recognized protocol type recognized in the case of IPv6 extension headers. This value defaults to 0xFF to indicate that no layer 4 header was recognized (possibly due to absence of an IP header).

Table 15-33. RQFPR Field Descriptions (continued)

PID <sup>1</sup>	Bit	Name	Description
1100	0–31	DIA	Destination IP address. If an IPv4 header was found, this is the entire destination address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit destination address. This value defaults to 0x0000_0000 if no IP header appeared.
1101	0–31	SIA	Source IP address. If an IPv4 header was found, this is the entire source address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit source address. This value defaults to 0x0000_0000 if no IP header appeared.
1110	0–15	—	Reserved, should be written with zero.
	16–31	DPT	Destination port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.
1111	0–15	—	Reserved, should be written with zero.
	16–31	SPT	Source port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.

<sup>1</sup> PID is the property identifier field of the filter table control entry (see RQFCR[PID]) at the same index.

### 15.5.3.3.9 Maximum Receive Buffer Length Register (MRBLR)

The MRBLR register is written by the user. It informs the eTSEC how much space is in the receive buffer pointed to by the RxBD. Figure 15-31 describes the definition for the MRBLR.



Figure 15-31. MRBLR Register Definition

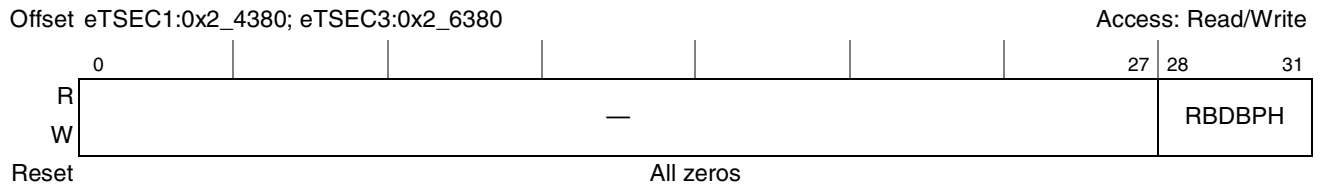
Table 15-34. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the eTSEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. The eTSEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL. MRBL must be set, together with the number of buffer descriptors, to ensure adequate space for received frames. See Section 15.5.3.5.5, “Maximum Frame Length Register (MAXFRM),” for further discussion.
26–31	—	To ensure that MRBL is a multiple of 64, these bits are reserved and should be cleared.

### 15.5.3.3.10 Receive Data Buffer Pointer High Register (RBDBPH)

The RBDBPH register is written by the user with the most significant address bits common to all RxBD buffer addresses, RxBD[Data Buffer Pointer]. As a consequence, Rx buffers must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBDBPH. The RxBD ring itself can

reside in a different memory region (based at RBASEH). Figure 15-32 describes the definition for the RBDBPH register.



**Figure 15-32. RBDBPH Register Definition**

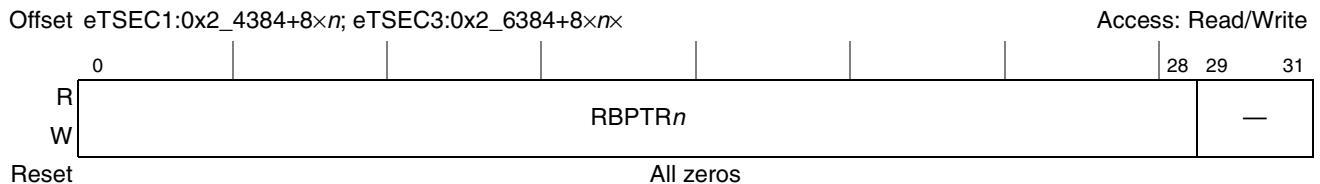
Table 15-35 describes the fields of the RBDBPH register.

**Table 15-35. RBDBPH Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	RBDBPH	Most significant bits common to all data buffer addresses contained in RxBDs. The user must initialize RBDBPH before enabling the eTSEC receive function.

### 15.5.3.3.11 Receive Buffer Descriptor Pointers 0–7 (RBPTR0–RBPTR7)

RBPTR0–RBPTR7 each contains the low-order 32 bits of the next receive buffer descriptor address for their respective RxBD ring. Figure 15-33 describes the RBPTR registers. These registers takes on the value of their ring’s associated RBASE when the RBASE register is written by software. Software must not write RBPTR $n$  while eTSEC is actively receiving frames. However, RBPTR $n$  can be modified when the receiver is disabled or when no Rx buffer is in use (after a GRACEFUL STOP RECEIVE command is issued and the frame completes its reception) in order to change the next RxBD eTSEC receives.



**Figure 15-33. RBPTR0–RBPTR7 Register Definition**

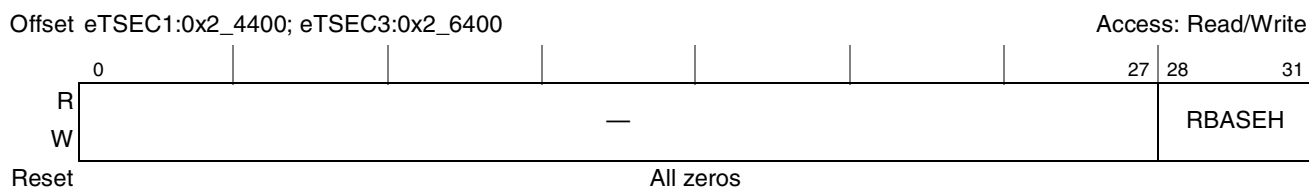
Table 15-23 describes the fields of the RBPTR $n$  register.

**Table 15-36. RBPTR $n$  Field Descriptions**

Bits	Name	Description
0–28	RBPTR $n$	Current RxBD pointer for RxBD ring $n$ . Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the RxBD ring is reached, eTSEC initializes RBPTR $n$ to the value in the corresponding RBASE $n$ . The RBPTR register is internally written by the eTSEC’s DMA controller during reception. The pointer increments by 8 (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved

### 15.5.3.3.12 Receive Descriptor Base Address High Register (RBASEH)

The RBASEH register is written by the user with the most significant address bits common to all RxB D addresses, including RBASE0–RBASE7 and RBPTR0–RBPTR7. As a consequence, RxB D rings must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBASEH. However, Rx data buffers may potentially reside in a different memory region based at RBDBPH. Figure 15-34 describes the definition for the RBASEH register.



**Figure 15-34. RBASEH Register Definition**

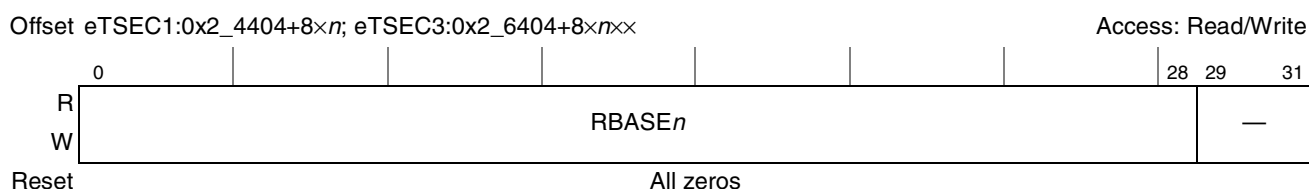
Table 15-24 describes the fields of the RBASEH register.

**Table 15-37. RBASEH Field Descriptions**

Bits	Name	Description
0–27	—	Reserved
28–31	RBASEH	Most significant bits common to all RxB D addresses—except data buffer pointers. The user must initialize RBASEH before enabling the eTSEC receive function.

### 15.5.3.3.13 Receive Descriptor Base Address Registers (RBASE0–RBASE7)

The RBASE $n$  registers are written by the user with the base address of each RxB D ring  $n$ . Each such value must be divisible by eight, since the 3 least-significant bits always write as 000. Figure 15-35 describes the RBASE $n$  registers.



**Figure 15-35. RBASE Register Definition**

Table 15-25 describes the fields of the RBASE $n$  registers.

**Table 15-38. RBASE0–RBASE7 Field Descriptions**

Bits	Name	Description
0–28	RBASE $n$	Receive base for ring $n$ . RBASE defines the starting location in the memory map for the eTSEC RxB Ds. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the receive packets. The user must initialize RBASE before enabling the eTSEC receive function on the associated ring.
29–31	—	Reserved

### 15.5.3.4 MAC Functionality

This section describes the MAC registers and provides a brief overview of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

#### 15.5.3.4.1 Configuring the MAC

The MAC configuration registers 1 and 2 provide for configuring the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value. Should custom preamble insertion/extraction be configured, then this register must be left at its default value.
- Varying pad/CRC combinations—Three different pad/CRC combinations are provided to handle a variety of system requirements. Simplest are frames that already have a valid frame check sequence (FCS) field. The other two options include appending a valid CRC or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

#### 15.5.3.4.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the eTSEC. Half-duplex mode is only supported for 10- and 100-Mbps operation. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

In full-duplex mode both the carrier sense (CRS) and collision (COL) indications from the PHY are ignored, but in half-duplex mode the eTSEC defers to CRS, and following a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if CRS is sensed. During the final one-third of the IPG, CRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

#### 15.5.3.4.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the eTSEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating that a collision occurred and that the start of the frame is needed for retransmission. The eTSEC then backs off of the medium for a time determined by the truncated binary exponential back off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured via the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system should flush the frame and move to the next one in line. If the system requests to send a packet while the eTSEC is deferring to a

carrier, the eTSEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

If packet transmission attempts experience collisions, the eTSEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the  $n$ th retransmission attempt is chosen as a uniformly-distributed random integer  $r$  in the range:

$$0 \leq r \leq 2^k, \text{ where } k = \min(n, 10).$$

So after the first collision, the eTSEC backs off either 0 or 1 slot times. After the fifth collision, the eTSEC backs off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back off is 1024. This can be adjusted via the half-duplex register. An alternate truncation point, such as 7 for instance, can be programmed. On average, the MAC is more aggressive after seven collisions than other stations on the network.

#### 15.5.3.4.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within eTSEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of the packet (including preamble), the end occurs around the 56th byte of the frame data. Slot time in 1000-Mbps mode is not supported.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The eTSEC implements the optional back pressure mechanism using the raise carrier method. If the system receive logic wishes to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is set (TCTRL[THDF]). If the medium is idle, the eTSEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, the eTSEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the back-pressure-no-back-off configuration bit HAFDUP[BP No BackOff]. These transmitting-preamble-for-back pressure collisions are not counted. If HAFDUP[BP No BackOff] is set, the eTSEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If HAFDUP[BP No BackOff] is cleared, the eTSEC adheres to the truncated BEB algorithm that allows the possibility of packets being received. This also can be detrimental in that packets can now experience excessive collisions, causing them to be dropped in the stations from which they originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, HAFDUP[BP No BackOff] must be set.

The eTSEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the eTSEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. HAFDUP[BP No BackOff] applies for any collision that occurs during the sending of this packet. Collisions for packets while half duplex back pressure is asserted are counted. The eTSEC does not defer while attempting to

send packets while in back pressure. Again, back pressure is non-standard, yet it can be effective in reducing the flow of receive packets.

### 15.5.3.4.5 Controlling PHY Links

Control and status to and from the PHY is provided via the two-wire MII management interface described in IEEE 802.3u. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) are used to exercise this interface between a host processor and one or more PHY devices (including the TBI).

The eTSEC MII's registers provide the ability to perform continuous read cycles (called a scan cycle); although, scan cycles are not explicitly defined in the standard. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY scan].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the eTSEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

### 15.5.3.5 MAC Registers

This section describes the MAC registers.

#### 15.5.3.5.1 MAC Configuration 1 Register (MACCFG1)

MACCFG1 is written by the user. Figure 15-36 describes the definition for the MACCFG1 register.

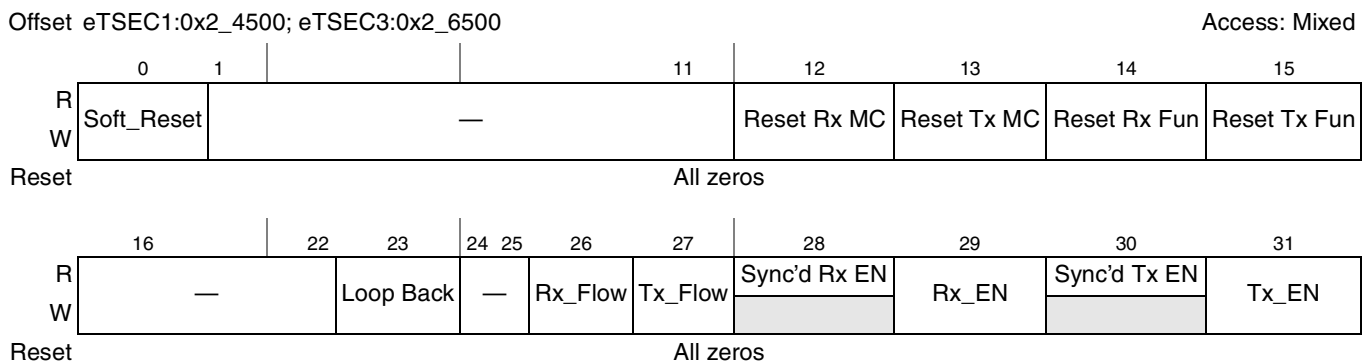


Figure 15-36. MACCFG1 Register Definition



Table 15-39 describes the fields of the MACCFG1 register.

**Table 15-39. MACCFG1 Field Descriptions**

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See <a href="#">Section 15.6.3.2, “Soft Reset and Reconfiguring Procedure,”</a> for more information on setting this bit. 0 Normal operation. 1 Place the entire MAC in reset except for the host interface.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the receive part of the MAC in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit part of the MAC in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation. 1 Place the receive function in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit function in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop Back	Loop back. This bit is cleared by default. 0 Normal operation. 1 Loop back the MAC transmit outputs to the MAC receive inputs.
24–25	—	Reserved
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream. (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.

**Table 15-39. MACCFG1 Field Descriptions (continued)**

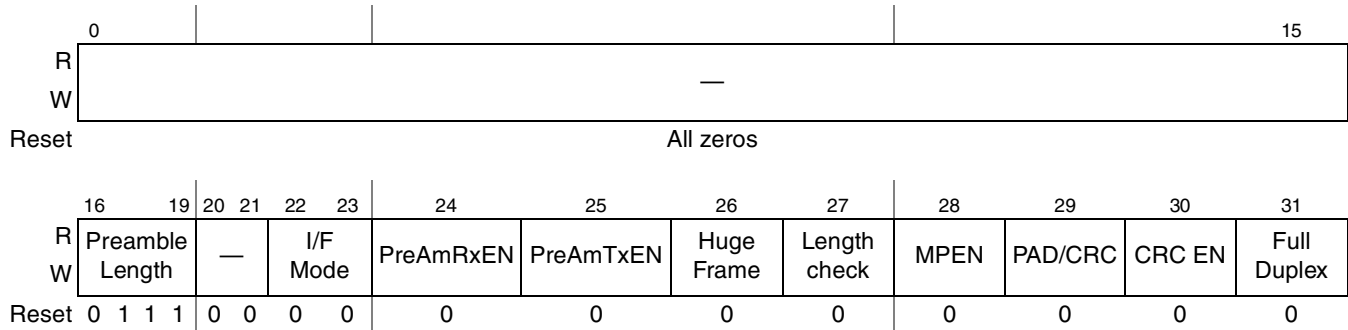
Bits	Name	Description
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream. (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

### 15.5.3.5.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. [Figure 15-37](#) describes the definition for the MACCFG2 register.

Offset eTSEC1:0x2\_4504; eTSEC3:0x2\_6504

Access: Read/Write

**Figure 15-37. MACCFG2 Register Definition**

[Table 15-40](#) describes the fields of the MACCFG2 register.

**Table 15-40. MACCFG2 Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble Length	This field determines the length in bytes of the preamble field preceding each Ethernet start-of-frame delimiter byte. Values from 0x3 to 0xF are supported by the controller. The default value of 0x7 should not be altered in order to guarantee reliable operation with IEEE 802.3 compliant hardware.
20–21	—	Reserved
22–23	I/F Mode	This field determines the type of interface to which the MAC is connected. Its default is 00. 00 Reserved bit mode (not supported) (10 Mbps GENDEC/GPSI) 01 Nibble mode (MII) (10/100 Mbps MII/RMII) 10 Byte mode (GMII/TBI) (1000 Mbps GMII/TBI). Reserved if neither GMII or TBI are supported. 11 Reserved
24	PreAM RxEN	User defined preamble enable for received frames. This bit is cleared by default. 0 The MAC skips the Ethernet preamble without returning it. 1 The MAC recovers the received Ethernet preamble and passes it to the driver at the start of each received frame. Not applicable to FIFO or RMII 10/100 modes.

Table 15-40. MACCFG2 Field Descriptions (continued)

Bits	Name	Description																				
25	PreAM TxEN	User defined preamble enable for transmitted frames. This bit is cleared by default. 0 The MAC generates a standard Ethernet preamble. 1 If a user-defined preamble has been passed to the MAC it is transmitted instead of the standard preamble. Otherwise the standard Ethernet preamble is generated. The Preamble Length field should be left at its default setting if a user-defined preamble is transmitted. Not applicable to FIFO or RMII 10/100 modes.																				
26	Huge Frame	Huge frame enable. This bit is cleared by default. 0 Limit the length of frames received to less than or equal to the maximum frame length value (MAXFRM[Maximum Frame]) and limit the length of frames transmitted to less than the maximum frame length. See Section 15.6.7, "Buffer Descriptors," for further details of buffer descriptor bit updating. <table border="1" data-bbox="456 634 1446 915"> <thead> <tr> <th>Frame type</th> <th>Frame length</th> <th>Packet truncation</th> <th>Buffer descriptor updated</th> </tr> </thead> <tbody> <tr> <td>Receive or transmit</td> <td>&gt; maximum frame length</td> <td>yes</td> <td>yes</td> </tr> <tr> <td>Receive</td> <td>= maximum frame length</td> <td>no</td> <td>yes</td> </tr> <tr> <td>Transmit</td> <td>= maximum frame length</td> <td>no</td> <td>no</td> </tr> <tr> <td>Receive or transmit</td> <td>&lt; maximum frame length</td> <td>no</td> <td>no</td> </tr> </tbody> </table> 1 Frames are transmitted and received regardless of their relationship to the maximum frame length. Note that if Huge Frame is cleared, the user must ensure that adequate buffer space is allocated for received frames. See Section 15.5.3.5.5, "Maximum Frame Length Register (MAXFRM)," for further information.	Frame type	Frame length	Packet truncation	Buffer descriptor updated	Receive or transmit	> maximum frame length	yes	yes	Receive	= maximum frame length	no	yes	Transmit	= maximum frame length	no	no	Receive or transmit	< maximum frame length	no	no
Frame type	Frame length	Packet truncation	Buffer descriptor updated																			
Receive or transmit	> maximum frame length	yes	yes																			
Receive	= maximum frame length	no	yes																			
Transmit	= maximum frame length	no	no																			
Receive or transmit	< maximum frame length	no	no																			
27	Length check	Length check. This bit is cleared by default. 0 No length field checking is performed. 1 The MAC checks the frame's length field on receive to ensure it matches the actual data field length. Transmitted frames are not checked.																				
28	MPEN	Magic packet enable for Ethernet modes. This bit is cleared by default. MPEN should be enabled only after GRACEFUL RECEIVE STOP and GRACEFUL TRANSMIT STOP are completed successfully (in other words, transmission and reception have stopped). 0 Normal receive behavior on receive, or Magic Packet mode has exited with reception of a valid Magic Packet. 1 Commence Magic Packet detection by the MAC provided that frame reception is enabled in MACCFG1. In this mode the MAC ignores all received frames until the specific Magic Packet frame is received, at which point this bit is cleared by the eTSEC, and a maskable interrupt via IEVENT[MAG] occurs.																				
29	PAD/CRC	Pad and append CRC. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.																				

**Table 15-40. MACCFG2 Field Descriptions (continued)**

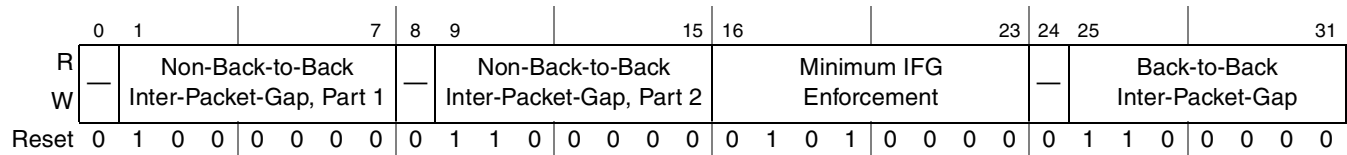
Bits	Name	Description
30	CRC EN	CRC enable. If the configuration bit PAD/CRC ENABLE or the per-packet PAD/CRC ENABLE is set, CRC ENABLE is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	Full Duplex	Full duplex configure. This bit is cleared by default. 0 The MAC operates in half-duplex mode only. 1 The MAC operates in full-duplex mode.

### 15.5.3.5.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. [Figure 15-38](#) describes the definition for IPGIFG.

Offset eTSEC1:0x2\_4508; eTSEC3:0x2\_6508

Access: Read/Write

**Figure 15-38. IPGIFG Register Definition**

[Table 15-41](#) describes the fields of the IPGIFG register.

**Table 15-41. IPGIFG Field Descriptions**

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	This is a programmable field representing the optional carrier sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	This is a programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	This is a programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	This is a programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

### 15.5.3.5.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 15-39 describes the HAFDUP register.

Offset eTSEC1:0x2\_450C; eTSEC3:0x2\_650C

Access: Read/Write

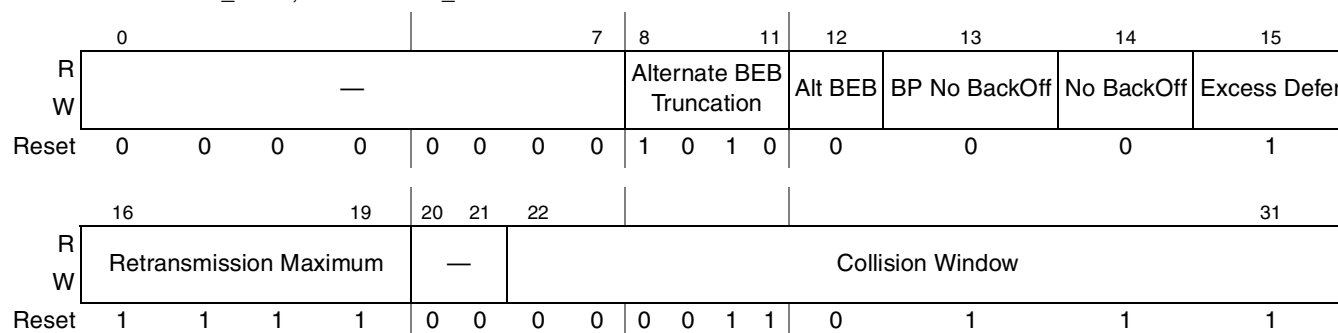


Figure 15-39. Half-Duplex Register Definition

Table 15-42 describes the fields of the HAFDUP register.

Table 15-42. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while ALTERNATE BINARY EXPONENTIAL BACKOFF ENABLE is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential backoff. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back off rule. 1 The Tx MAC uses the ALTERNATE BINARY EXPONENTIAL BACKOFF TRUNCATION setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum backoff time.
13	BP No BackOff	Back pressure no backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.
14	No BackOff	No backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF.
20–21	—	Reserved
22–31	Collision Window	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

### 15.5.3.5.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 15-40 shows the MAXFRM register.

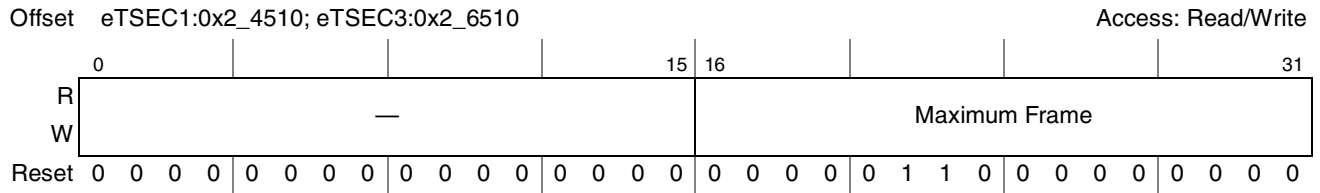


Figure 15-40. Maximum Frame Length Register Definition

Table 15-43 describes the fields of the MAXFRM register.

Table 15-43. MAXFRM Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536 bytes). It sets the maximum Ethernet frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].). It does not affect the size of packets sent or received via the FIFO packet interface.  Note that if MACCFG2[Huge Frame] = 0, the value of this field must be less than or equal to MRBLR[MRBL] × (minimum number of RxBDs per ring). See Section 15.5.3.5.2, “MAC Configuration 2 Register (MACCFG2),” Section 15.5.3.3.9, “Maximum Receive Buffer Length Register (MRBLR),” and Section 15.6.7.3, “Receive Buffer Descriptors (RxBd).”

### 15.5.3.5.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user to configure all MII management operations. Note that MII management hardware is shared by all eTSECs. Thus, only through the MIIM registers of eTSEC1 can external PHYs be accessed and configured. Note: when an eTSEC is configured to use TBI/RTBI, configuration of the TBI/RTBI (described in Section 15.5.4, “Ten-Bit Interface (TBI)”) is done via the MIIM registers for that eTSEC. For example, if a TBI/RTBI interface is required on eTSEC3, then the MIIM registers starting at offset 0x2\_6520 are used to configure it.

Figure 15-41 describes the definition for the MIIMCFG register.

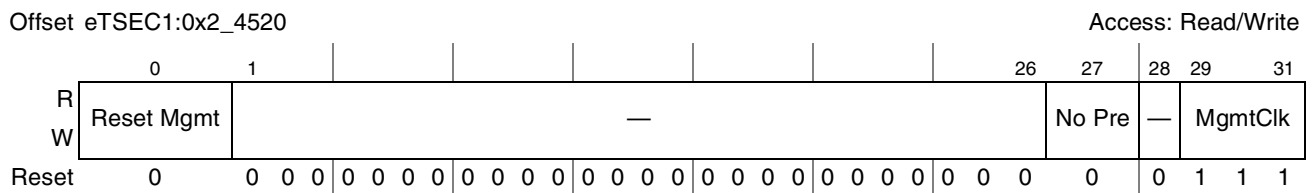


Figure 15-41. MII Management Configuration Register Definition

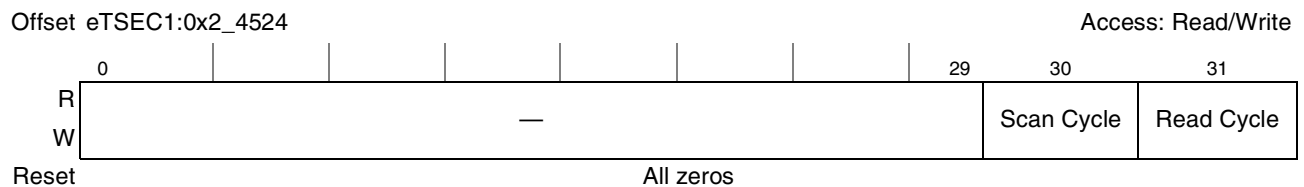
Table 15-44 describes the fields of the MIIMCFG register.

**Table 15-44. MIIMCFG Field Descriptions**

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII MGMT to perform mgmt read/write cycles if requested via the host interface. 1 Reset the MII MGMT.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII MGMT performs Mgmt read/write cycles with 32 clocks of preamble. 1 The MII MGMT suppresses preamble generation and reduces the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	MgmtClk	This field determines the clock frequency of the MII management clock (EC_MDC). Its default value is 111. <b>Note:</b> The eTSEC system clock is derived from (CCB Clock)/2. 000 1/4 of the eTSEC system clock divided by 8 001 1/4 of the eTSEC system clock divided by 8 010 1/6 of the eTSEC system clock divided by 8 011 1/8 of the eTSEC system clock divided by 8 100 1/10 of the eTSEC system clock divided by 8 101 1/14 of the eTSEC system clock divided by 8 110 1/20 of the eTSEC system clock divided by 8 111 1/28 of the eTSEC system clock divided by 8

### 15.5.3.5.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 15-42 describes the definition for MIIMCOM.



**Figure 15-42. MIIMCOM Register Definition**

Table 15-45 describes the fields of the MIIMCOM register.

**Table 15-45. MIIMCOM Descriptions**

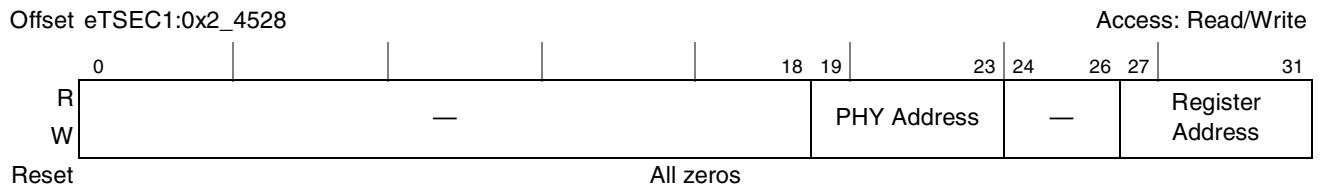
Bits	Name	Description
0–29	—	Reserved

**Table 15-45. MIIMCOM Descriptions (continued)**

Bits	Name	Description
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation. 1 The MII management continuously performs read cycles. This is useful for monitoring link fail, for example.
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation. 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0-to-1 transition of this bit also causes the MIIMIND[Busy] bit to be set. The read is complete when the MIIMIND[Busy] bit clears. Data is returned in register MIIMSTAT[PHY Status].

**15.5.3.5.8 MII Management Address Register (MIIMADD)**

The MIIMADD register is written by the user. [Figure 15-43](#) shows the MIIMADD register.



**Figure 15-43. MIIMADD Register Definition**

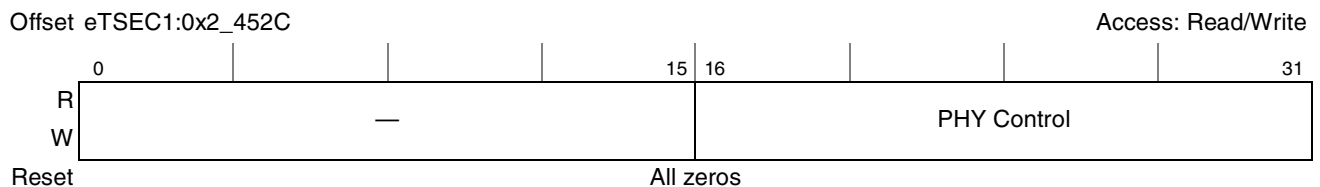
[Table 15-46](#) describes the fields of the MIIMADD register.

**Table 15-46. MIIMADD Field Descriptions**

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved). Its default value is 0x00.
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of Mgmt cycles. Up to 32 registers can be accessed. Its default value is 0x00.

**15.5.3.5.9 MII Management Control Register (MIIMCON)**

MIIMCON, shown in [Figure 15-44](#), is written by the user.



**Figure 15-44. MII Mgmt Control Register Definition**



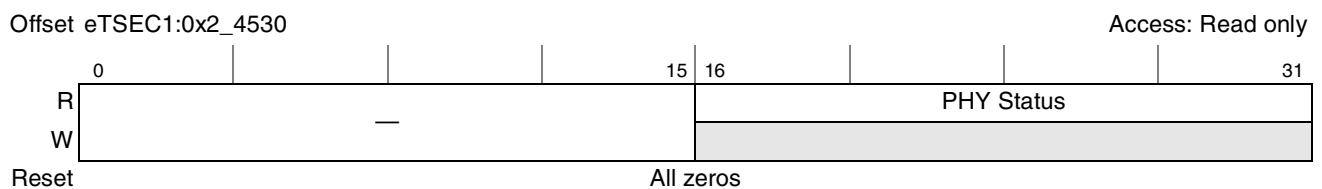
Table 15-47 describes the fields of the MIIMCON register.

**Table 15-47. MIIMCON Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Control	If written, an MII Mgmt write cycle is performed using this 16-bit data, the pre-configured PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). Its default value is 0x0000.

### 15.5.3.5.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read only by the user. Figure 15-45 describes the definition for the MIIMSTAT register.



**Figure 15-45. MIIMSTAT Register Definition**

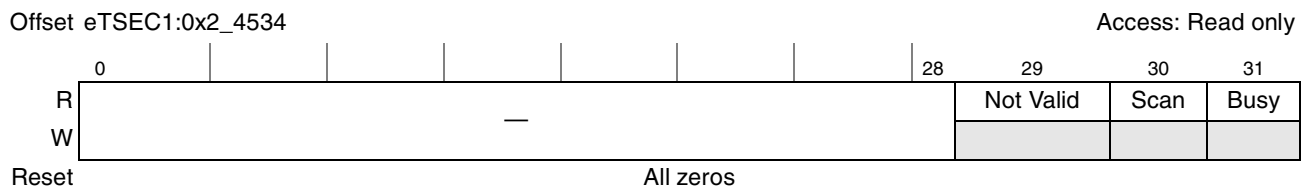
Table 15-48 describes the fields of the MIIMSTAT register.

**Table 15-48. MIIMSTAT Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII Mgmt read cycle, the 16-bit data can be read from this location. Its default value is 0x0000.

### 15.5.3.5.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 15-46 describes the definition for the MIIMIND register.



**Figure 15-46. MII Mgmt Indicator Register Definition**

**Table 15-49. MIIMIND Field Descriptions**

Bits	Name	Description
0–28	—	Reserved
29	Not Valid	Not valid. 0 MII Mgmt read cycle has completed and the read data is valid. 1 MII Mgmt read cycle has not completed and the read data is not yet valid.
30	Scan	Scan in progress. 0 A scan operation (continuous MII Mgmt read cycles) is not in progress. 1 A scan operation (continuous MII Mgmt read cycles) is in progress.
31	Busy	Busy. 0 MII Mgmt block is not currently performing an MII Mgmt read or write cycle. 1 MII Mgmt block is currently performing an MII Mgmt read or write cycle.

### 15.5.3.5.12 Interface Status Register (IFSTAT)

Figure 15-47 shows the IFSTAT register.

Offset eTSEC1:0x2\_453C; eTSEC3:0x2\_653C

Access: Read only

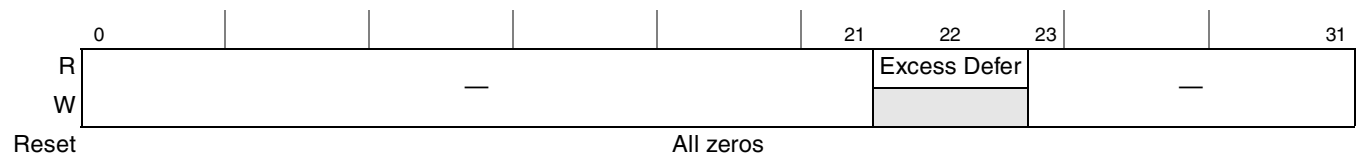
**Figure 15-47. Interface Status Register Definition**

Table 15-50 describes the fields of the FSTAT register.

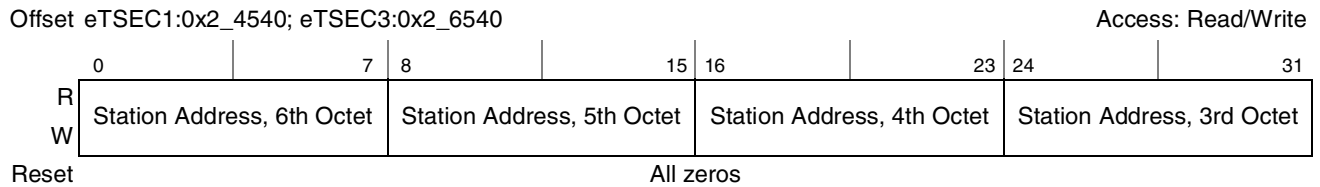
**Table 15-50. IFSTAT Field Descriptions**

Bits	Name	Description
0–21	—	Reserved
22	Excess Defer	Excessive transmission defer. This bit latches high and is cleared when read. This bit is cleared by default. 0 Normal operation. 1 The MAC excessively defers a transmission.
23–31	—	Reserved

### 15.5.3.5.13 MAC Station Address Part 1 Register (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. The value of the station address written into MACSTNADDR1 and MACSTNADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, MACSTNADDR1 is set to 0xCDAB7856 and MACSTNADDR2 is set to 0x34120000.

Figure 15-48 shows the MACSTNADDR1 register.



**Figure 15-48. MAC Station Address Part 1 Register Definition**

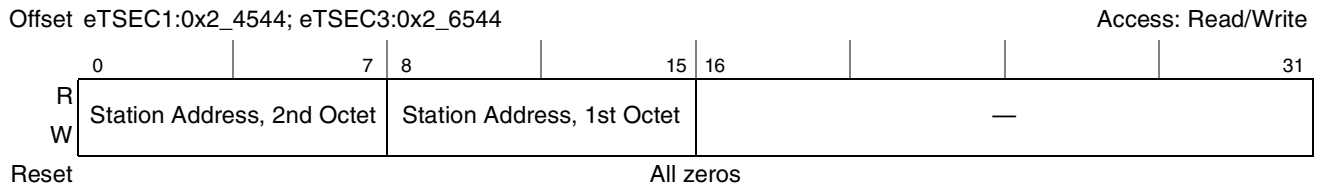
Table 15-51 describes the fields of the MACSTNADDR1 register.

**Table 15-51. MACSTNADDR1 Field Descriptions**

Bit	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x0.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x0.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x0.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x0.

#### 15.5.3.5.14 MAC Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 15-49 describes the definition for the MACSTNADDR2 register.



**Figure 15-49. MAC Station Address Part 2 Register Definition**

Table 15-52 describes the fields of the MACSTNADDR2 register.

**Table 15-52. MACSTNADDR2 Field Descriptions**

Bit	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x0.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

### 15.5.3.5.15 MAC Exact Match Address 1–15 Part 1 Registers (MAC01ADDR1–MAC15ADDR1)

The MAC01ADDR1–MAC15ADDR1 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 15-50 describes the definition for all of the fifteen MAC<sub>n</sub>ADDR1 registers. The value of the address written into MAC<sub>x</sub>ADDR1 and MAC<sub>n</sub>ADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a MAC address of 0x12345678ABCD, MAC<sub>n</sub>ADDR1 is set to 0xCDAB7856 and MAC<sub>n</sub>ADDR2 is set to 0x34120000. For any valid, non-zero MAC address received, exact match registers can be excluded individually by clearing them to all zero bytes.

Offset eTSEC1:0x2\_4548+8×*n*; eTSEC3:0x2\_6548+8×*n*

Access: Read/Write

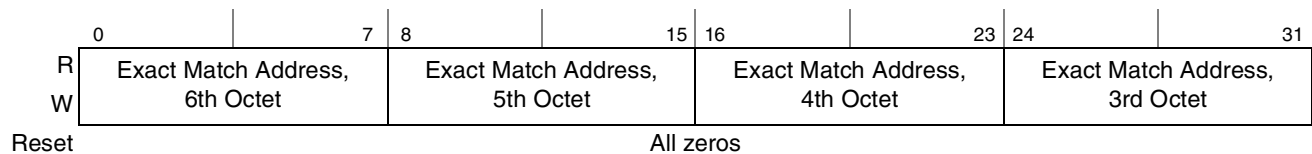


Figure 15-50. MAC Exact Match Address *n* Part 1 Register Definition

Table 15-51 describes the fields of a MAC<sub>n</sub>ADDR1 register.

Table 15-53. MAC<sub>n</sub>ADDR1 Field Descriptions

Bit	Name	Description
0–7	Exact Match Address, 6th Octet	Holds the sixth octet of the exact match address. The sixth octet (destination address bits 40–47) defaults to a value of 0x0.
8–15	Exact Match Address, 5th Octet	Holds the fifth octet of the exact match address. The fifth octet (destination address bits 32–39) defaults to a value of 0x0.
16–23	Exact Match Address, 4th Octet	Holds the fourth octet of the exact match address. The fourth octet (destination address bits 24–31) defaults to a value of 0x0.
24–31	Exact Match Address, 3rd Octet	Holds the third octet of the exact match address. The third octet (destination address bits 16–23) defaults to a value of 0x0.

### 15.5.3.5.16 MAC Exact Match Address 1–15 Part 2 Registers (MAC01ADDR2–MAC15ADDR2)

The MAC01ADDR2–MAC15ADDR2 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 15-51 describes the definition for all of the fifteen MAC<sub>x</sub>ADDR2 registers.

Offset eTSEC1:0x2\_454C+8×*n*; eTSEC3:0x2\_654C+8×*n*

Access: Read/Write

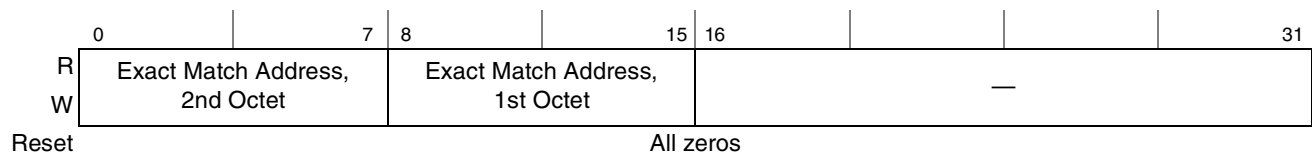


Figure 15-51. MAC Exact Match Address *x* Part 2 Register Definition

Table 15-52 describes the fields of a MACxADDR2 register.

**Table 15-54. MAC01ADDR2–MAC15ADDR2 Field Descriptions**

Bit	Name	Description
0–7	Exact Match Address, 2nd Octet	This field holds the second octet of the exact match address. The second octet (destination address bits 8–15) defaults to a value of 0x0.
8–15	Exact Match Address, 1st Octet	This field holds the first octet of the exact match address. The first octet (destination address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

### 15.5.3.6 MIB Registers

This section describes the MIB registers. The eTSEC RMON module has 37 separate statistics counters, which simply count or accumulate statistical events that occur as packets transmitted and received. These counters support RMON MIB group 1, RMON MIB group 2 if table counters, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB.

An interrupt can be generated upon any one counter's rollover condition via a carry interrupt output from the RMON. Each counter's rollover condition can be discretely masked from causing an interrupt by internal masking registers. In addition, each individual counter value may be reset on read access, or all counters may be simultaneously reset by setting ECNTRL[CLRCNT].

The majority of MIB counters are Ethernet-specific.

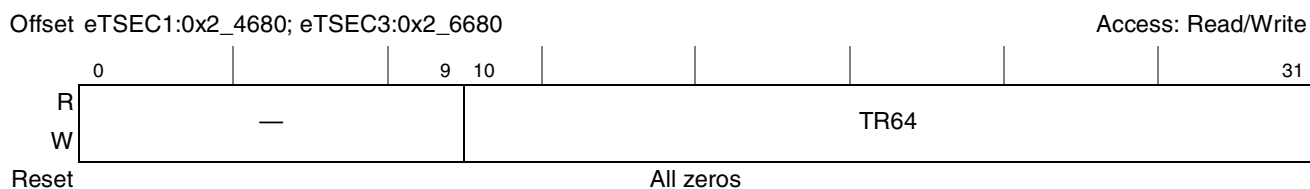
In FIFO modes, only the following registers are updated:

- Transmit: TBYT, TPKT, TDRP
- Receive: RBYT, RPKT, RFCS

**Note:** RMON counters will not comprehend custom VLAN tagged frames. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF. Specifically, custom VLAN tagged frames will not be afforded the ability to be greater than 1518, as the IEEE standard tagged frames will be.

#### 15.5.3.6.1 Transmit and Receive 64-Byte Frame Counter (TR64)

Figure 15-52 describes the definition for the TR64 register.



**Figure 15-52. Transmit and Receive 64-Byte Frame Register Definition**

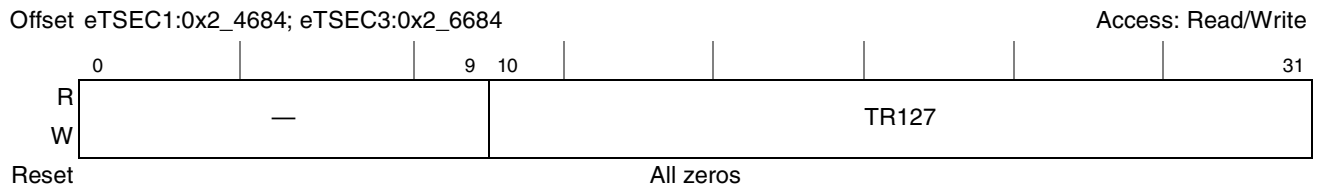
Table 15-55 describes the fields of the TR64 register.

**Table 15-55. TR64 Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TR64	Transmit and receive 64-byte frame counter—Increment for each good or bad frame transmitted and received which is 64 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.2 Transmit and Receive 65- to 127-Byte Frame Counter (TR127)

Figure 15-53 describes the definition for the TR127 register.



**Figure 15-53. Transmit and Receive 65- to 127-Byte Frame Register Definition**

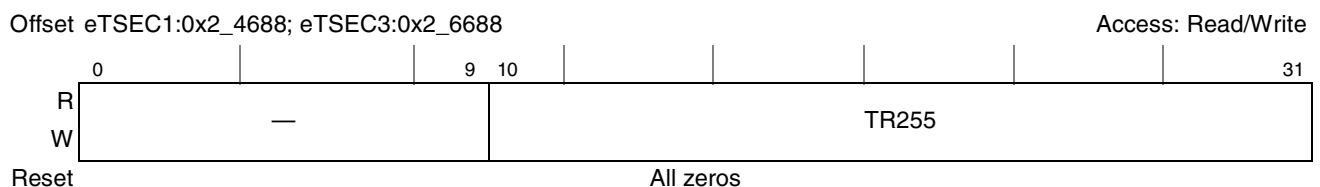
Table 15-56 describes the fields of the TR127 register.

**Table 15-56. TR127 Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TR127	Transmit and receive 65- to 127-byte frame counter—Increments for each good or bad frame transmitted and received which is 65–127 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.3 Transmit and Receive 128- to 255-Byte Frame Counter (TR255)

Figure 15-54 describes the definition for the TR255 register.



**Figure 15-54. Transmit and Received 128- to 255-Byte Frame Register Definition**

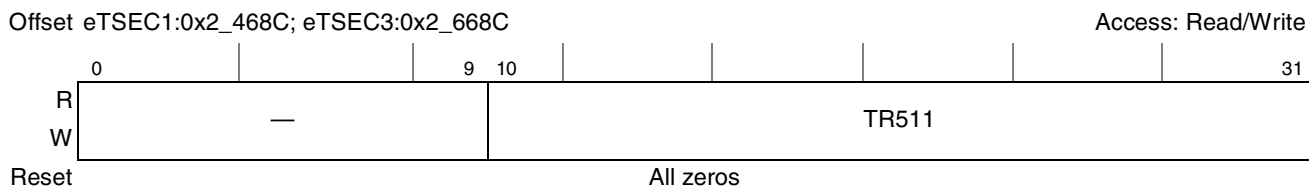
Table 15-57 describes the fields of the TR255 register.

**Table 15-57. TR255 Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TR255	Transmit and receive 128- to 255-byte frame counter—Increments for each good or bad frame transmitted and received which is 128–255 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.4 Transmit and Receive 256- to 511-Byte Frame Counter (TR511)

Figure 15-55 describes the definition for the TR511 register.



**Figure 15-55. Transmit and Received 256- to 511-Byte Frame Register Definition**

Table 15-58 describes the fields of the TR511 register.

**Table 15-58. TR511 Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TR511	Increments for each good or bad frame transmitted and received which is 256–511 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.5 Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)

Figure 15-56 shows the TR1K register.



**Figure 15-56. Transmit and Received 512- to 1023-Byte Frame Register Definition**

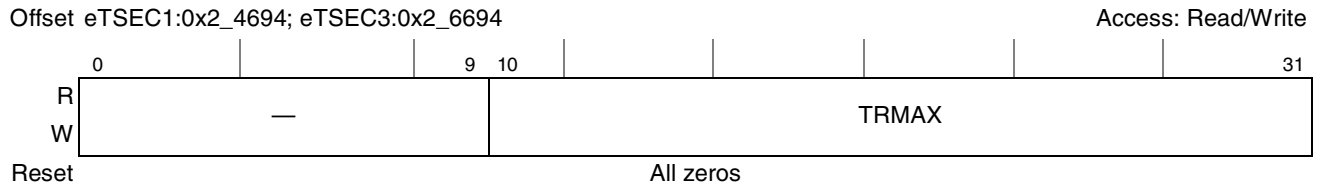
Table 15-59 describes the fields of the TR1K register.

**Table 15-59. TR1K Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TR1K	Increments for each good or bad frame transmitted and received which is 512–1023 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.6 Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)

Figure 15-57 describes the definition for the TRMAX register.



**Figure 15-57. Transmit and Received 1024- to 1518-Byte Frame Register Definition**

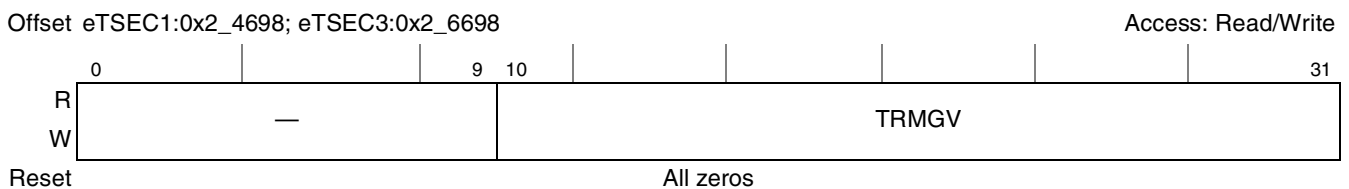
Table 15-60 describes the fields of the TRMAX register.

**Table 15-60. TRMAX Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TRMAX	Increments for each good or bad frame transmitted and received which is 1024–1518 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

### 15.5.3.6.7 Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)

Figure 15-58 describes the definition for the TRMGV register.



**Figure 15-58. Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition**

Table 15-61 describes the fields of the TRMGV register.

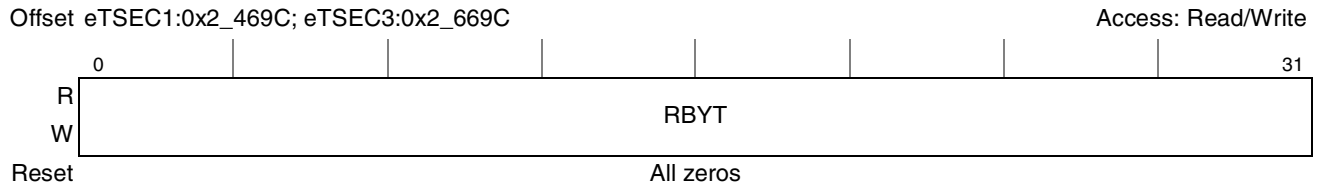
**Table 15-61. TRMGV Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TRMGV	Increments for each good or bad frame transmitted and received which is 1519–1522 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).



### 15.5.3.6.8 Receive Byte Counter (RBYT)

Figure 15-59 shows the RBYT register.



**Figure 15-59. Receive Byte Counter Register Definition**

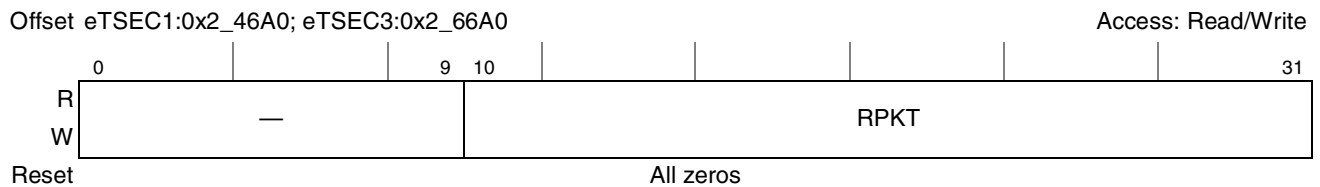
Table 15-62 describes the fields of the RBYT register.

**Table 15-62. RBYT Field Descriptions**

Bits	Name	Description
0–31	RBYT	Receive byte counter. The statistic counter register increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes. In FIFO mode, all bytes (including FCS bytes) are counted.

### 15.5.3.6.9 Receive Packet Counter (RPKT)

Figure 15-60 describes the definition for the RPKT register.



**Figure 15-60. Receive Packet Counter Register Definition**

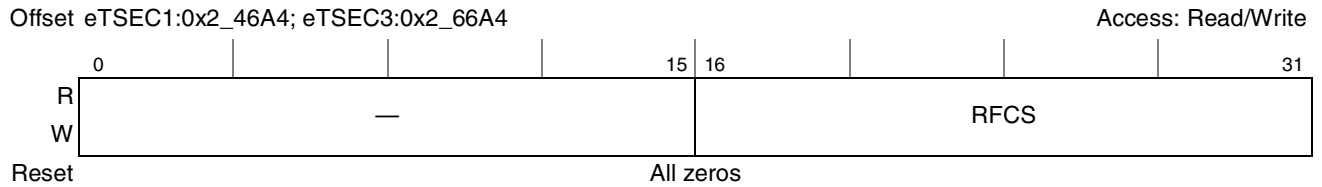
Table 15-63 describes the fields of the RPKT register.

**Table 15-63. RPKT Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	RPKT	Receive packet counter. Increments for each frame received packet (including bad packets, all unicast, broadcast, and multicast packets).

### 15.5.3.6.10 Receive FCS Error Counter (RFCS)

Figure 15-61 describes the definition for the RFCS register.



**Figure 15-61. Receive FCS Error Counter Register Definition**

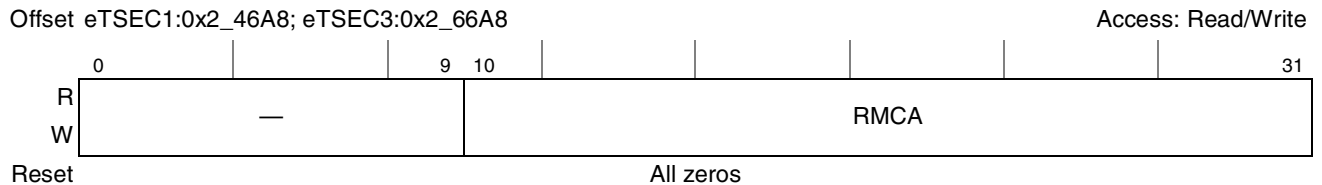
Table 15-64 describes the fields of the RFCS register.

**Table 15-64. RFCS Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RFCS	Receive FCS error counter. In Ethernet mode, increments for each frame received that has an integral 64–1518 length and contains a frame check sequence error. In FIFO mode, increments for each frame received that contains a frame check sequence error (regardless of size).

### 15.5.3.6.11 Receive Multicast Packet Counter (RMCA)

Figure 15-62 describes the definition for the RMCA register.



**Figure 15-62. Receive Multicast Packet Counter Register Definition**

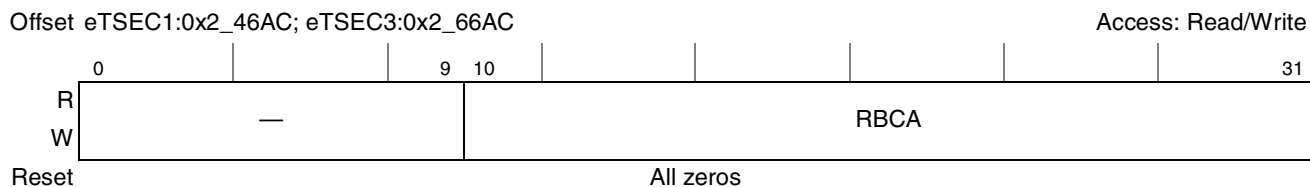
Table 15-65 describes the fields of the RMCA register.

**Table 15-65. RMCA Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	RMCA	Receive multicast packet counter. Increments for each multicast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

### 15.5.3.6.12 Receive Broadcast Packet Counter (RBCA)

Figure 15-63 describes the definition for the RBCA register.



**Figure 15-63. Receive Broadcast Packet Counter Register Definition**

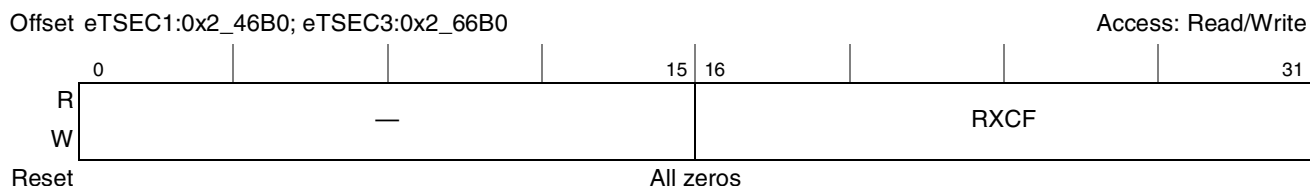
Table 15-66 describes the fields of the RBCA register.

**Table 15-66. RBCA Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	RBCA	Receive broadcast packet counter. Increments for each broadcast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

### 15.5.3.6.13 Receive Control Frame Packet Counter (RXCF)

Figure 15-64 describes the definition for the RXCF register.



**Figure 15-64. Receive Control Frame Packet Counter Register Definition**

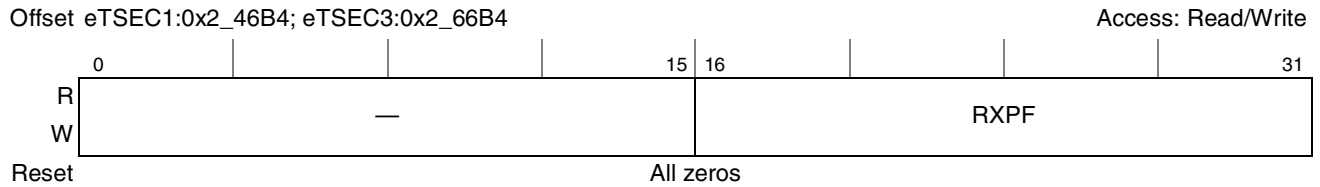
Table 15-67 describes the fields of the RXCF register.

**Table 15-67. RXCF Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RXCF	Receive control frame packet counter. Increments for each MAC control frame received (PAUSE and unsupported) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.14 Receive Pause Frame Packet Counter (RXPF)

Figure 15-65 describes the definition for the RXPF register.



**Figure 15-65. Receive Pause Frame Packet Counter Register Definition**

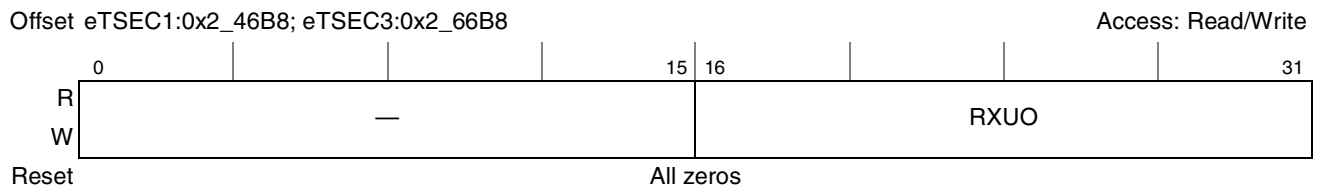
Table 15-68 describes the fields of the RXPF register.

**Table 15-68. RXPF Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RXPF	Receive PAUSE frame packet counter. Increments each time a PAUSE MAC control frame is received with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.15 Receive Unknown Opcode Packet Counter (RXUO)

Figure 15-66 describes the definition for the RXUO register.



**Figure 15-66. Receive Unknown Opcode Packet Counter Register Definition**

Table 15-69 describes the fields of the RXUO register.

**Table 15-69. RXUO Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RXUO	Receive unknown opcode counter. Increments each time a MAC control frame is received which contains an opcode other than PAUSE, but the frame has valid CRC and length 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.16 Receive Alignment Error Counter (RALN)

Figure 15-67 describes the definition for the RALN register.



**Figure 15-67. Receive Alignment Error Counter Register Definition**

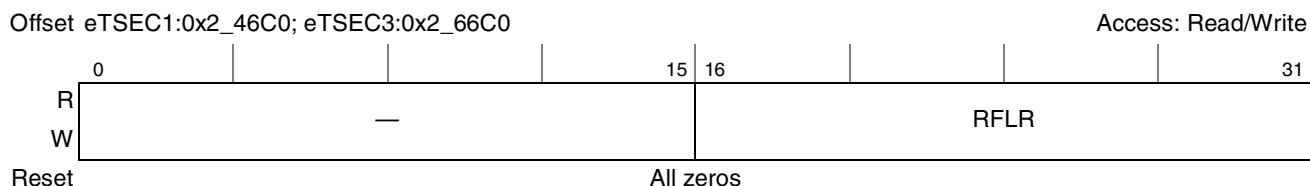
Table 15-70 describes the fields of the RALN register.

**Table 15-70. RALN Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RALN	Receive alignment error counter. Increments for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.

### 15.5.3.6.17 Receive Frame Length Error Counter (RFLR)

Figure 15-68 describes the definition for the RFLR register.



**Figure 15-68. Receive Frame Length Error Counter Register Definition**

Table 15-71 describes the fields of the RFLR register.

**Table 15-71. RFLR Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RFLR	Receive frame length error counter. Increments for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46–1500 bytes). The counter does not increment if the length field is not a valid 802.3 length, such as an Ethertype value.

### 15.5.3.6.18 Receive Code Error Counter (RCDE)

Figure 15-69 describes the definition for the RCDE register.

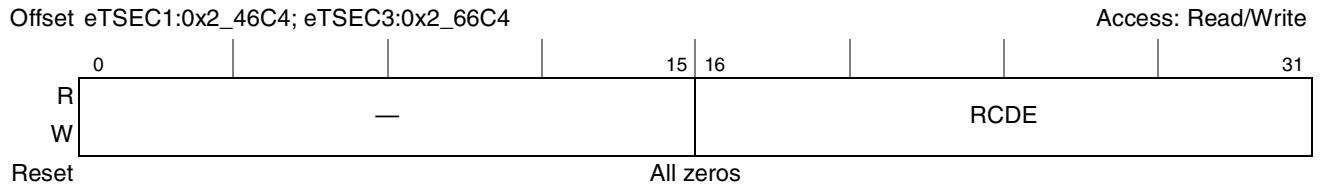


Figure 15-69. Receive Code Error Counter Register Definition

Table 15-72 describes the fields of the RCDE register.

Table 15-72. RCDE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCDE	Receive code error counter. Increments each time a valid carrier is present and at least one invalid data symbol is detected.

### 15.5.3.6.19 Receive Carrier Sense Error Counter (RCSE)

Figure 15-70 describes the definition for the RCSE register.

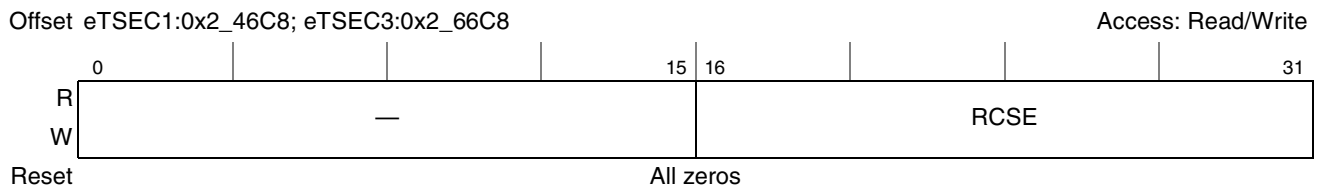


Figure 15-70. Receive Carrier Sense Error Counter Register Definition

Table 15-73 describes the fields of the RCSE register.

Table 15-73. RCSE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCSE	Receive false carrier counter. Counts the number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface. The count represented by an instance of this object is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt. The event is reported along with the statistics generated on the next received frame, as defined by a 1 on TSEC <sub>n</sub> _RX_ER and an 0xE on TSEC <sub>n</sub> _RXD. Only one false carrier condition can be detected and logged between frames.

### 15.5.3.6.20 Receive Undersize Packet Counter (RUND)

Figure 15-71 describes the definition for the RUND register.



**Figure 15-71. Receive Undersize Packet Counter Register Definition**

Table 15-74 describes the fields of the RUND register.

**Table 15-74. RUND Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RUND	Receive undersize packet counter. Increments each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This count does not include range length errors.

### 15.5.3.6.21 Receive Oversize Packet Counter (ROVR)

Figure 15-72 describes the definition for the ROVR register.



**Figure 15-72. Receive Oversize Packet Counter Register Definition**

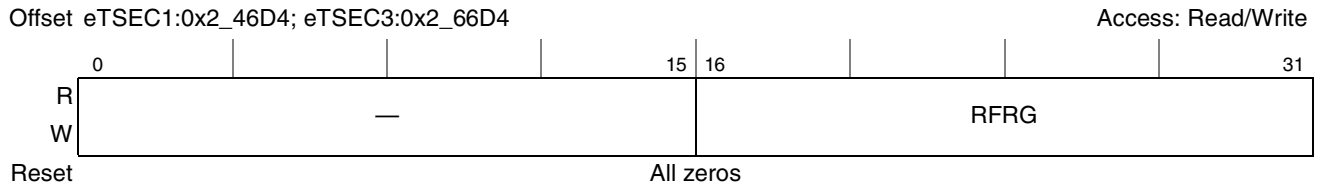
Table 15-75 describes the fields of the ROVR register.

**Table 15-75. ROVR Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	ROVR	Receive oversize packet counter. Increments each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and was otherwise well formed.

### 15.5.3.6.22 Receive Fragments Counter (RFRG)

Figure 15-73 describes the definition for the RFRG register.



**Figure 15-73. Receive Fragments Counter Register Definition**

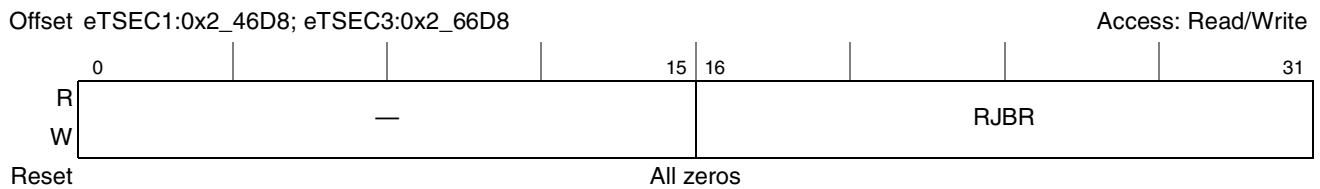
Table 15-76 describes the fields of the RFRG register.

**Table 15-76. RFRG Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RFRG	Receive fragments counter. Increments for each frame received which is less than 64 bytes in length and contains an invalid FCS. This includes integral and non-integral lengths.

### 15.5.3.6.23 Receive Jabber Counter (RJBR)

Figure 15-74 describes the definition for the RJBR register.



**Figure 15-74. Receive Jabber Counter Register Definition**

Table 15-77 describes the fields of the RJBR register.

**Table 15-77. RJBR Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RJBR	Receive jabber counter. Increments for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contain an invalid FCS. This includes alignment errors.



### 15.5.3.6.24 Receive Dropped Packet Counter (RDRP)

Figure 15-75 describes the definition for the RDRP register.



Figure 15-75. Receive Dropped Packet Counter Register Definition

Table 15-78 describes the fields of the RDRP register.

Table 15-78. RDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RDRP	Receive dropped packets counter. Increments for frames received which are streamed to system but are later dropped due to lack of system resources.

### 15.5.3.6.25 Transmit Byte Counter (TBYT)

Figure 15-76 describes the definition for the TBYT register.

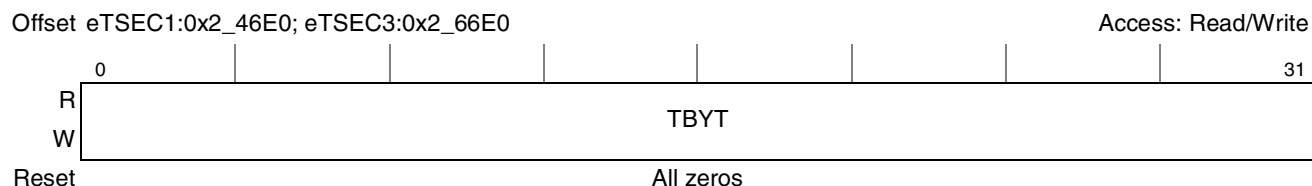


Figure 15-76. Transmit Byte Counter Register Definition

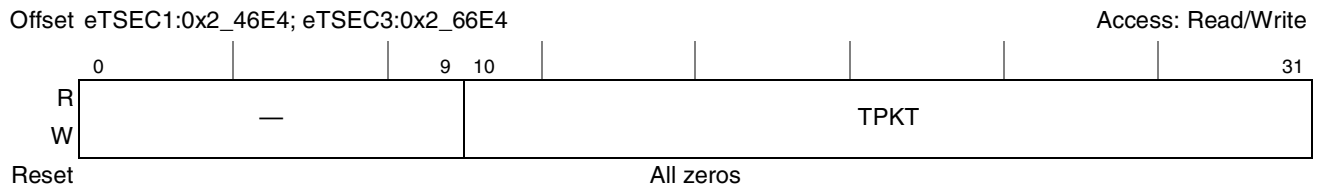
Table 15-79 describes the fields of the TBYT register.

Table 15-79. TBYT Field Descriptions

Bits	Name	Description
0–31	TBYT	Transmit byte counter. Increments by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes. Note that the value of TBYT may be greater than the actual number of bytes transmitted if the frame is truncated because it exceeds MAXFRM.

### 15.5.3.6.26 Transmit Packet Counter (TPKT)

Figure 15-77 describes the definition for the TPKT register.



**Figure 15-77. Transmit Packet Counter Register Definition**

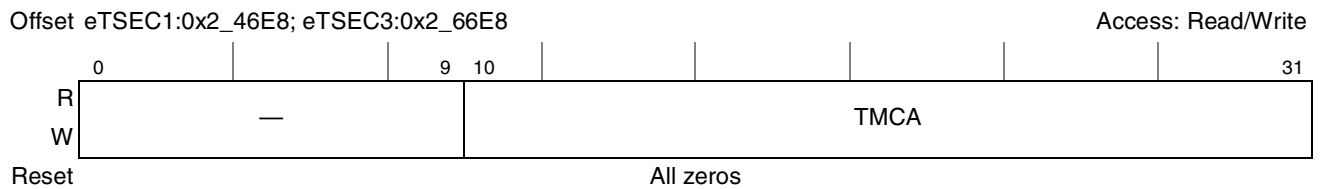
Table 15-80 describes the fields of the TPKT register.

**Table 15-80. TPKT Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TPKT	Transmit packet counter. Increments for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all unicast, broadcast, and multicast packets).

### 15.5.3.6.27 Transmit Multicast Packet Counter (TMCA)

Figure 15-78 describes the definition for the TMCA register.



**Figure 15-78. Transmit Multicast Packet Counter Register Definition**

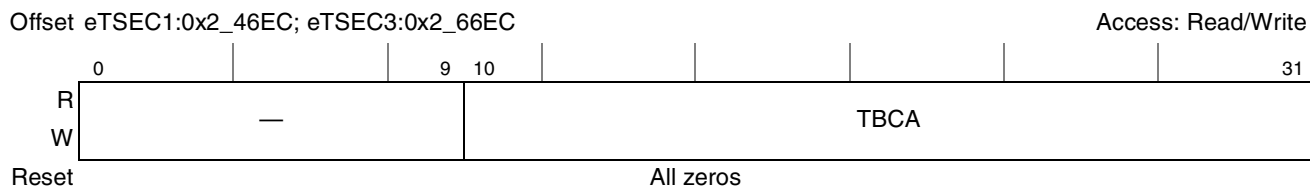
Table 15-81 describes the fields of the TMCA register.

**Table 15-81. TMCA Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TMCA	Transmit multicast packet counter. Increments for each multicast valid frame transmitted (excluding broadcast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.28 Transmit Broadcast Packet Counter (TBCA)

Figure 15-79 describes the definition for the TBCA register.



**Figure 15-79. Transmit Broadcast Packet Counter Register Definition**

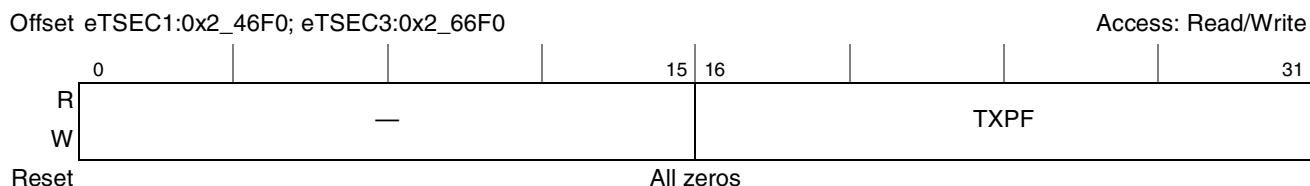
Table 15-82 describes the fields of the TBCA register.

**Table 15-82. TBCA Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	TBCA	Transmit broadcast packet counter. Increments for each broadcast frame transmitted (excluding multicast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.29 Transmit Pause Control Frame Counter (TXPF)

Figure 15-80 describes the definition for the TXPF register.



**Figure 15-80. Transmit Pause Control Frame Counter Register Definition**

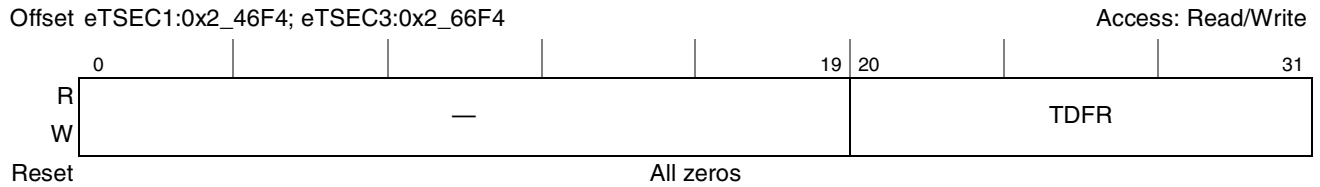
Table 15-83 describes the fields of the TXPF register.

**Table 15-83. TXPF Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	TXPF	Transmit PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is transmitted with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.30 Transmit Deferral Packet Counter (TDFR)

Figure 15-81 describes the definition for the TDFR register.



**Figure 15-81. Transmit Deferral Packet Counter Register Definition**

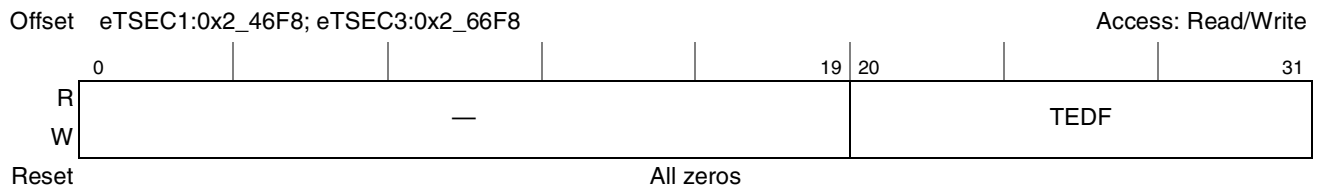
Table 15-84 describes the fields of the TDFR register.

**Table 15-84. TDFR Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit deferral packet counter. Increments for each frame, which was deferred on its first transmission attempt. This count does not include frames involved in collisions.

### 15.5.3.6.31 Transmit Excessive Deferral Packet Counter (TEDF)

Figure 15-82 describes the definition for the TEDF register.



**Figure 15-82. Transmit Excessive Deferral Packet Counter Register Definition**

Table 15-85 describes the fields of the TEDF register.

**Table 15-85. TEDF Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TEDF	Transmit excessive deferral packet counter. Increments for frames aborted which were deferred for an excessive period of time (3036 byte times).

### 15.5.3.6.32 Transmit Single Collision Packet Counter (TSCL)

Figure 15-83 describes the definition for the TSCL register.

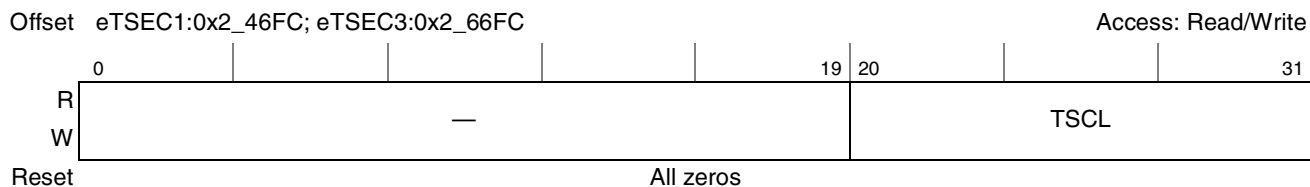


Figure 15-83. Transmit Single Collision Packet Counter Register Definition

Table 15-86 describes the fields of the TSCL register.

Table 15-86. TSCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TSCL	Transmit single collision packet counter. Increments for each frame transmitted which experienced exactly one collision during transmission.

### 15.5.3.6.33 Transmit Multiple Collision Packet Counter (TMCL)

Figure 15-84 describes the definition for the TMCL register.



Figure 15-84. Transmit Multiple Collision Packet Counter Register Definition

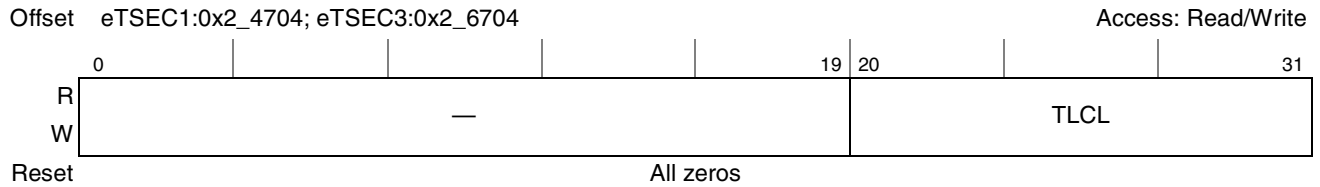
Table 15-87 describes the fields of the TMCL register.

Table 15-87. TMCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TMCL	Transmit multiple collision packet counter. Increments for each frame transmitted which experienced 2–15 collisions (including any late collisions) during transmission as defined using the Half_Duplex[RETRANSMISSION MAXIMUM] field.

### 15.5.3.6.34 Transmit Late Collision Packet Counter (TLCL)

Figure 15-85 describes the definition for the TLCL register.



**Figure 15-85. Transmit Late Collision Packet Counter Register Definition**

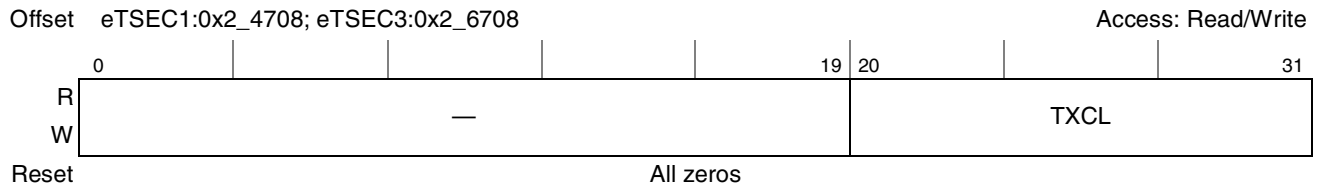
Table 15-88 describes the fields of the TLCL register.

**Table 15-88. TLCL Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TLCL	Transmit late collision packet counter. Increments for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the collision window field of the half-duplex [26:31] register.

### 15.5.3.6.35 Transmit Excessive Collision Packet Counter (TXCL)

Figure 15-86 describes the definition for the TXCL register.



**Figure 15-86. Transmit Excessive Collision Packet Counter Register Definition**

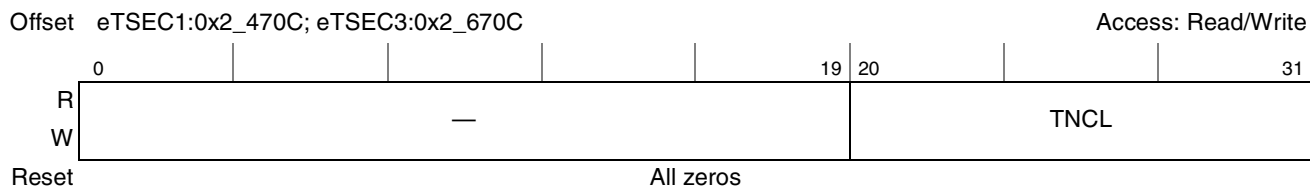
Table 15-89 describes the fields of the TXCL register.

**Table 15-89. TXCL Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TXCL	Transmit excessive collision packet counter. Increments for each frame that experienced 16 collisions during transmission and was aborted.

### 15.5.3.6.36 Transmit Total Collision Counter (TNCL)

Figure 15-87 describes the definition for the TNCL register.



**Figure 15-87. Transmit Total Collision Counter Register Definition**

Table 15-90 describes the fields of the TNCL register.

**Table 15-90. TNCL Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TNCL	Transmit total collision counter. Increments by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (That is, transmitting and receiving at the same time). <b>Note:</b> This count does not include collisions that result in an excessive collision condition.

### 15.5.3.6.37 Transmit Drop Frame Counter (TDRP)

Figure 15-88 describes the definition for the TDRP register.



**Figure 15-88. Transmit Drop Frame Counter Register Definition**

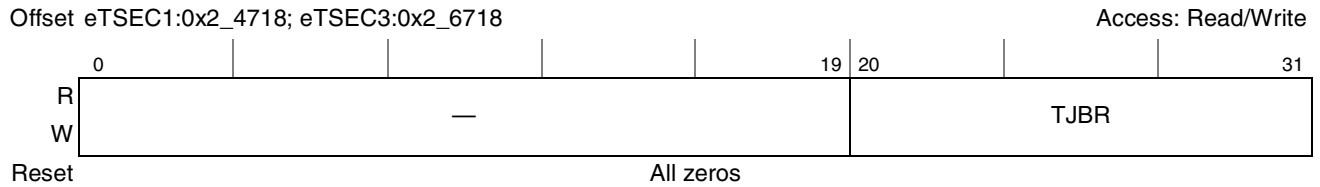
Table 15-91 describes the fields of the TDRP register.

**Table 15-91. TDRP Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time a memory error or an underrun has occurred.

### 15.5.3.6.38 Transmit Jabber Frame Counter (TJBR)

Figure 15-89 describes the definition for the TJBR register.



**Figure 15-89. Transmit Jabber Frame Counter Register Definition**

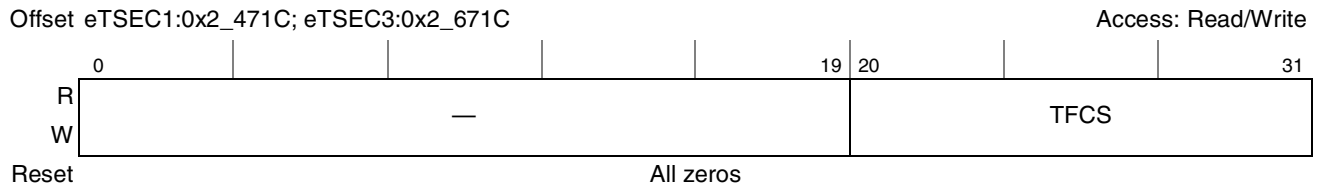
Table 15-92 describes the fields of the TJBR register.

**Table 15-92. TJBR Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TJBR	Transmit jabber frame counter. Increments for each oversized transmitted frame with an incorrect FCS value.

### 15.5.3.6.39 Transmit FCS Error Counter (TFCS)

Figure 15-90 describes the definition for the TFCS register.



**Figure 15-90. Transmit FCS Error Counter Register Definition**

Table 15-93 describes the fields of the TFCS register.

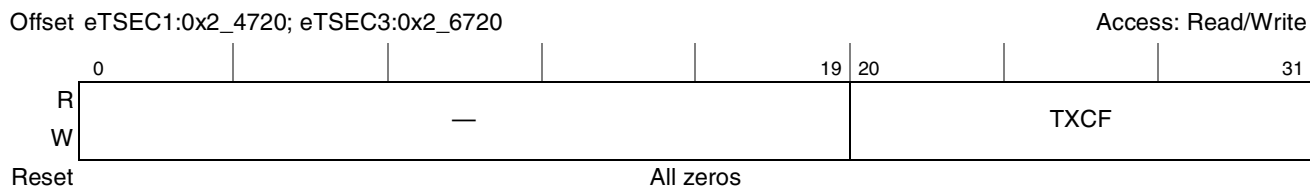
**Table 15-93. TFCS Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TFCS	Transmit FCS error counter. Increments for every valid sized packet with an incorrect FCS value.



### 15.5.3.6.40 Transmit Control Frame Counter (TXCF)

Figure 15-91 describes the definition for the TXCF register.



**Figure 15-91. Transmit Control Frame Counter Register Definition**

Table 15-94 describes the fields of the TXCF register.

**Table 15-94. TXCF Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TXCF	Transmit control frame counter. Increments for every control frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

### 15.5.3.6.41 Transmit Oversize Frame Counter (TOVR)

Figure 15-92 describes the definition for the TOVR register.



**Figure 15-92. Transmit Oversized Frame Counter Register Definition**

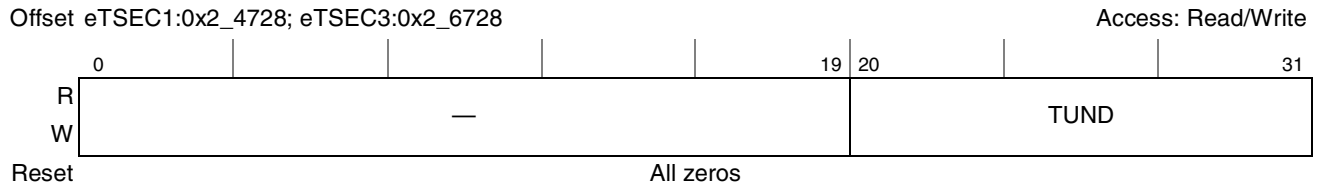
Table 15-95 describes the fields of the TOVR register.

**Table 15-95. TOVR Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TOVR	Transmit oversize frame counter. Increments for each oversized transmitted frame with a correct FCS value.

### 15.5.3.6.42 Transmit Undersize Frame Counter (TUND)

Figure 15-93 describes the definition for the TUND register.



**Figure 15-93. Transmit Undersize Frame Counter Register Definition**

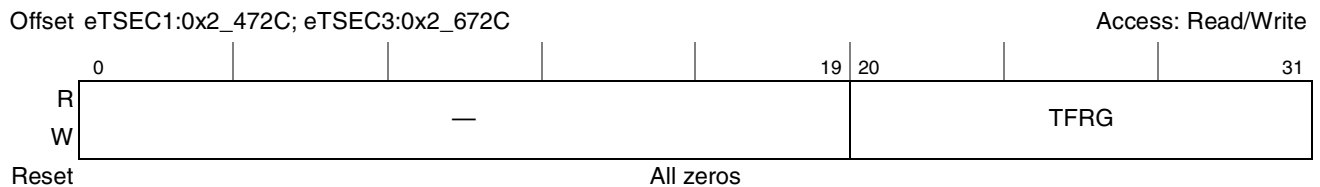
Table 15-96 describes the fields of the TUND register.

**Table 15-96. TUND Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TUND	Transmit undersize frame counter. Increments for every frame less than 64 bytes, with a correct FCS value.

### 15.5.3.6.43 Transmit Fragment Counter (TFRG)

Figure 15-94 describes the definition for the TFRG register.



**Figure 15-94. Transmit Fragment Counter Register Definition**

Table 15-97 describes the fields of the TFRG register.

**Table 15-97. TFRG Field Descriptions**

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less than 64 bytes, with an incorrect FCS value.

### 15.5.3.6.44 Carry Register 1 (CAR1)

Carry register bits are cleared on carry register writes when the respective bits are set. Figure 15-95 describes the definition for the CAR1 register.

Offset eTSEC1:0x2\_4730; eTSEC3:0x2\_6730

Access: Read/Write

	0	1	2	3	4	5	6	7						13	14	15
R	C1	C1	C1	C1	C1	C1	C1		—						C1	C1
W	64	127	255	511	1K	MAX	MGV								REJ	RBY
Reset	All zeros															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1
W	RPK	RFC	RMC	RBC	RXC	RXP	R XU	RAL	RFL	RCD	RCS	RUN	ROV	RFR	RJB	RDR
Reset	All zeros															

Figure 15-95. Carry Register 1 (CAR1) Register Definition

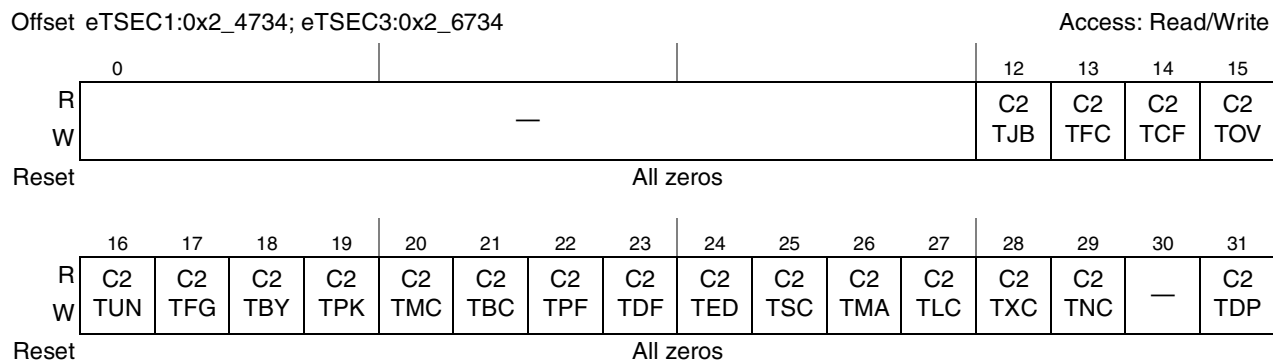
Table 15-98 describes the fields of the CAR1 register.

**Table 15-98. CAR1 Field Descriptions**

Bits	Name	Description
0	C164	Carry register 1 TR64 counter carry bit
1	C1127	Carry register 1 TR127 counter carry bit
2	C1255	Carry register 1 TR255 counter carry bit
3	C1511	Carry register 1 TR511 counter carry bit
4	C11K	Carry register 1 TR1K counter carry bit
5	C1MAX	Carry register 1 TRMAX counter carry bit
6	C1MGV	Carry register 1 TRMGV counter carry bit
7–13	—	Reserved
14	C1REJ	Carry register 1 RREJ counter carry bit
15	C1RBY	Carry register 1 RBYT counter carry bit
16	C1RPK	Carry register 1 RPKT counter carry bit
17	C1RFC	Carry register 1 RFCS counter carry bit
18	C1RMC	Carry register 1 RMCA counter carry bit
19	C1RBC	Carry register 1 RBCA counter carry bit
20	C1RXC	Carry register 1 RXCF counter carry bit
21	C1RXP	Carry register 1 RXPf counter carry bit
22	C1RXU	Carry register 1 RXUO counter carry bit
23	C1RAL	Carry register 1 RALN counter carry bit
24	C1RFL	Carry register 1 RFLR counter carry bit
25	C1RCD	Carry register 1 RCDE counter carry bit
26	C1RCS	Carry register 1 RCSE counter carry bit
27	C1RUN	Carry register 1 RUND counter carry bit
28	C1ROV	Carry register 1 ROVR counter carry bit
29	C1RFR	Carry register 1 RFRG counter carry bit
30	C1RJB	Carry register 1 RJBR counter carry bit
31	C1RDR	Carry register 1 RDRP counter carry bit

### 15.5.3.6.45 Carry Register 2 (CAR2)

Figure 15-96 describes the definition for the CAR2 register.



**Figure 15-96. Carry Register 2 (CAR2) Register Definition**

Carry register bits are cleared on carry register write when the respective bits are set. Table 15-99 describes the fields of the CAR2 register.

**Table 15-99. CAR2 Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12	C2TJB	Carry register 2 TJBR counter carry bit
13	C2TFC	Carry register 2 TFCS counter carry bit
14	C2TCF	Carry register 2 TXCF counter carry bit
15	C2TOV	Carry register 2 TOVR counter carry bit
16	C2TUN	Carry register 2 TUND counter carry bit
17	C2TFG	Carry register 2 TFRG counter carry bit
18	C2TBY	Carry register 2 TBYT counter carry bit
19	C2TPK	Carry register 2 TPKT counter carry bit
20	C2TMC	Carry register 2 TMCA counter carry bit
21	C2TBC	Carry register 2 TBCA counter carry bit
22	C2TPF	Carry register 2 TXPF counter carry bit
23	C2TDF	Carry register 2 TDFR counter carry bit
24	C2TED	Carry register 2 TEDF counter carry bit
25	C2TSC	Carry register 2 TSCL counter carry bit
26	C2TMA	Carry register 2 TMCL counter carry bit
27	C2TLC	Carry register 2 TLCL counter carry bit
28	C2TXC	Carry register 2 TXCL counter carry bit
29	C2TNC	Carry register 2 TNCL counter carry bit

**Table 15-99. CAR2 Field Descriptions (continued)**

Bits	Name	Description
30	—	Reserved, should be cleared
31	C2TDP	Carry register 2 TDRP counter carry bit

### 15.5.3.6.46 Carry Mask Register 1 (CAM1)

While one of the below mask bits are cleared, the corresponding carry bit in CAR1 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits all default to a set state. [Figure 15-97](#) describes the definition for the CAM1 register.

Offset eTSEC1:0x2\_4738; eTSEC3:0x2\_6738 Access: Read/Write

	0	1	2	3	4	5	6	7					13	14	15	
R	M1	M1	M1	M1	M1	M1	M1							M1	M1	
W	64	127	255	511	1K	MAX	MGV							REJ	RBY	
Reset	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1
W	RPK	RFC	RMC	RBC	RXC	RXP	R XU	RAL	RFL	RCD	RCS	RUN	ROV	RFR	RJB	RDR
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

**Figure 15-97. Carry Mask Register 1 (CAM1) Register Definition**

[Table 15-100](#) describes the fields of the CAM1 register.

**Table 15-100. CAM1 Field Descriptions**

Bits	Name	Description
0	M164	Mask register 1 TR64 counter carry bit mask
1	M1127	Mask register 1 TR127 counter carry bit mask
2	M1255	Mask register 1 TR255 counter carry bit mask
3	M1511	Mask register 1 TR511 counter carry bit mask
4	M11k	Mask register 1 TR1K counter carry bit mask
5	M1MAX	Mask register 1 TRMAX counter carry bit mask
6	M1MGV	Mask register 1 TRMGV counter carry bit mask
7–13	—	Reserved
14	M1REJ	Mask register 1 RREJ counter carry bit mask
15	M1RBY	Mask register 1 RBYT counter carry bit mask
16	M1RPK	Mask register 1 RPKT counter carry bit mask
17	M1RFC	Mask register 1 RFCS counter carry bit mask
18	M1RMC	Mask register 1 RMCA counter carry bit mask
19	M1RBC	Mask register 1 RBCA counter carry bit mask

Table 15-100. CAM1 Field Descriptions (continued)

Bits	Name	Description
20	M1RXC	Mask register 1 RXCF counter carry bit mask
21	M1RXP	Mask register 1 RXPf counter carry bit mask
22	M1RXU	Mask register 1 RXUO counter carry bit mask
23	M1RAL	Mask register 1 RALN counter carry bit mask
24	M1RFL	Mask register 1 RFLR counter carry bit mask
25	M1RCD	Mask register 1 RCDE counter carry bit mask
26	M1RCS	Mask register 1 RCSE counter carry bit mask
27	M1RUN	Mask register 1 RUND counter carry bit mask
28	M1ROV	Mask register 1 ROVR counter carry bit mask
29	M1RFR	Mask register 1 RFRG counter carry bit mask
30	M1RJB	Mask register 1 RJBR counter carry bit mask
31	M1RDR	Mask register 1 RDRP counter carry bit mask

### 15.5.3.6.47 Carry Mask Register 2 (CAM2)

While one of the below mask bits are cleared, the corresponding carry bit in CAR2 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits default to a set state. Figure 15-98 describes the definition for the CAM2 register.

Offset eTSEC1:0x2\_473C; eTSEC3:0x2\_673C

Access: Read/Write

	0									11	12	13	14	15		
R																
W	—															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	—	M2
W	TUN	TFG	TBY	TPK	TMC	TBC	TPF	TDF	TED	TSC	TMA	TLC	TXC	TNC		TDP
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Figure 15-98. Carry Mask Register 2 (CAM2) Register Definition

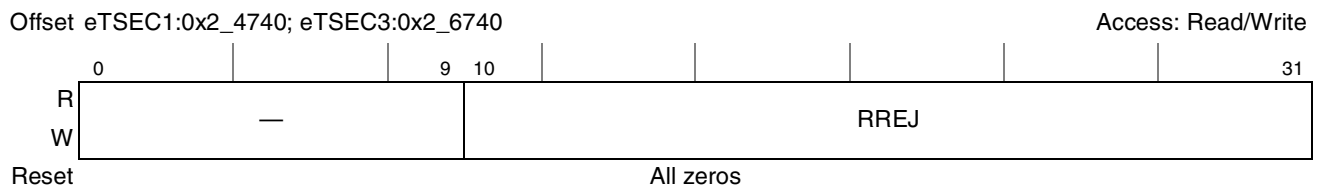
Table 15-101 describes the fields of the CAM2 register.

**Table 15-101. CAM2 Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12	M2TJB	Mask register 2 TJBR counter carry bit mask
13	M2TFC	Mask register 2 TFCS counter carry bit mask
14	M2TCF	Mask register 2 TXCF counter carry bit mask
15	M2TOV	Mask register 2 TOVR counter carry bit mask
16	M2TUN	Mask register 2 TUND counter carry bit mask
17	M2TFG	Mask register 2 TFRG counter carry bit mask
18	M2TBY	Mask register 2 TBYT counter carry bit mask
19	M2TPK	Mask register 2 TPKT counter carry bit mask
20	M2TMC	Mask register 2 TMCA counter carry bit mask
21	M2TBC	Mask register 2 TBCA counter carry bit mask
22	M2TPF	Mask register 2 TXPF counter carry bit mask
23	M2TDF	Mask register 2 TDFR counter carry bit mask
24	M2TED	Mask register 2 TEDF counter carry bit mask
25	M2TSC	Mask register 2 TSCL counter carry bit mask
26	M2TMA	Mask register 2 TMCL counter carry bit mask
27	M2TLC	Mask register 2 TLCL counter carry bit mask
28	M2TXC	Mask register 2 TXCL counter carry bit mask
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved
31	M2TDP	Mask register 2 TDRP counter carry bit mask

**15.5.3.6.48 Receive Filer Rejected Packet Counter (RREJ)**

Figure 15-99 describes the definition for the RREJ register.



**Figure 15-99. Receive Filer Rejected Packet Counter Register Definition**



Table 15-66 describes the fields of the RREJ register.

**Table 15-102. RREJ Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10–31	RREJ	Receive filter rejected packet counter. Increments for each frame with valid CRC received, but rejected by the receive queue filter—either due to a matching rule that asserted the REJ flag or due to filtering to a RxBD ring that was not enabled (see IEVENT[FIQ] error).

### 15.5.3.7 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. The DA field of every received frame is processed through a 32-bit CRC generator (CRC-32 polynomial), and the 8 or 9 most significant bits of the CRC are mapped to a hash table entry. The user can enable a hash entry by setting its bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Software may need to further filter the address in order to eliminate false-positive hits in the hash table.

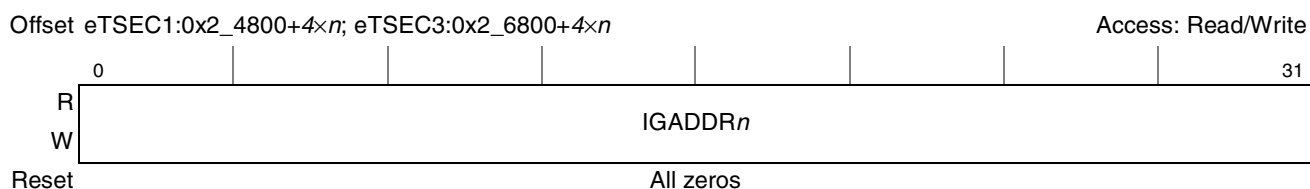
If  $RCTRL[GHTX] = 0$ , the 8 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 comprise a 256-entry hash table exclusively for individual (unicast) address matching, while registers GADDR0–GADDR7 comprise a 256-entry hash table for group (multicast) address matching. If  $RCTRL[GHTX] = 1$ , the group hash table is extended to all 512 entries, and the 9 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 hold hash table entries 0–255 for group addresses, while registers GADDR0–GADDR7 hold entries 256–511 of the extended group hash table.

See Section 15.6.3.7.2, “Hash Table Algorithm” for more information on the hash algorithm.

#### 15.5.3.7.1 Individual/Group Address Registers 0–7 (IGADDR $n$ )

The IGADDR $n$  registers are written by the user. Together these registers represent, depending on  $RCTRL[GHTX]$ , either the 256 entries of the individual address hash table, or the first 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC-32 result points to an enabled hash entry.

Figure 15-100 describes the definition for the IGADDR $n$  register.



**Figure 15-100. IGADDR $n$  Register Definition**

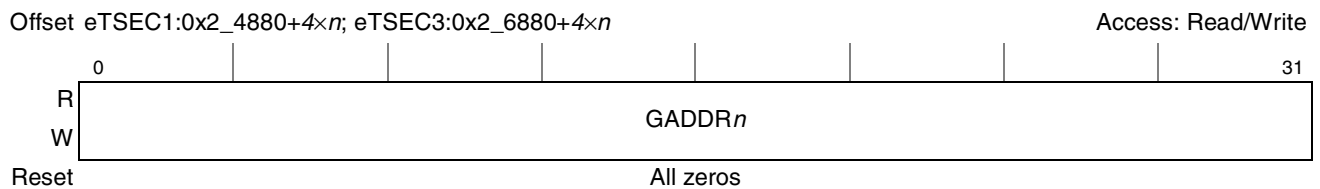
Table 15-104 describes the fields of the IGADDR $n$  register.

**Table 15-103. IGADDR $n$  Field Descriptions**

Bits	Name	Description
0–31	IGADDR $n$	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, IGADDR0 contains entries 0–31 of the 256-entry individual hash table and IGADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, IGADDR0 contains entries 0–31 of the 512-entry extended group hash table and IGADDR7 represents entries 224–255.

### 15.5.3.7.2 Group Address Registers 0–7 (GADDR $n$ )

The GADDR $n$  registers are written by the user. Together these registers represent, depending on RCTRL[GHTX], either the 256 entries of the group address hash table, or the last 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 15-101 describes the definition for the GADDR $n$  register.



**Figure 15-101. GADDR $n$  Register Definition**

Table 15-104 describes the fields of the GADDR $n$  register.

**Table 15-104. GADDR $n$  Field Descriptions**

Bits	Name	Description
0–31	GADDR $n$	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, GADDR0 contains entries 0–31 of the 256-entry group hash table and GADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, GADDR0 contains entries 256–287 of the 512-entry extended group hash table and GADDR7 represents entries 480–511.

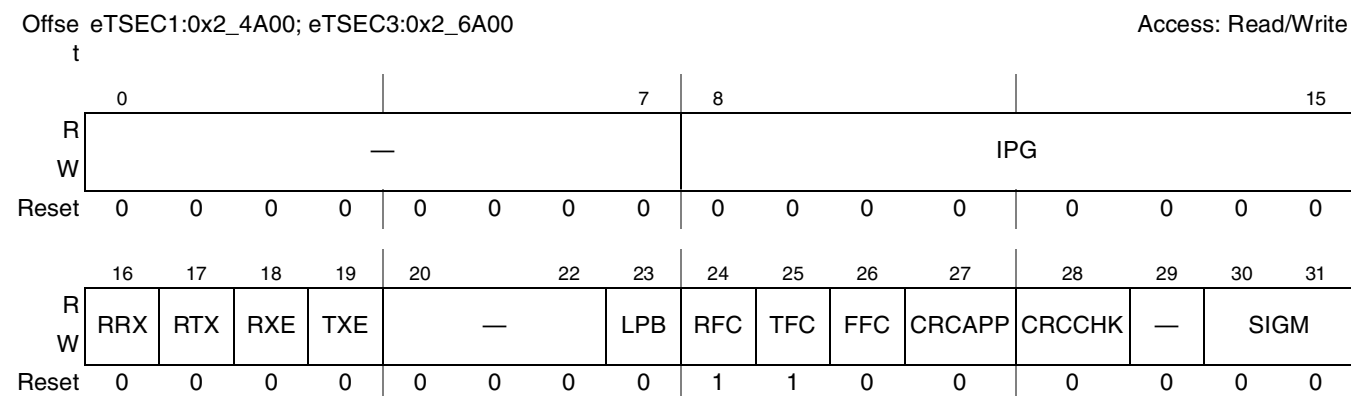
### 15.5.3.8 FIFO Registers

This section provides detailed descriptions of the registers used to configure the FIFO interface. All of the registers are 32 bits wide. The ECNTRL[FIFM] bit is set to indicate that data transfers take place over this interface. Please refer to [Section 15.6.2, “Connecting to FIFO Interfaces,”](#) for details of the signaling protocols available.

#### 15.5.3.8.1 FIFO Configuration Register (FIFOCFG)

The FIFO Configuration Register configures and enables the 8-bit FIFO interface.

[Figure 15-102](#) describes the definition for the FIFOCFG register.



**Figure 15-102. FIFOCFG Register Definition**

[Table 15-105](#) describes the fields of the FIFOCFG register.

**Table 15-105. FIFOCFG Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–15	IPG	Minimum inter packet gap. This sets the minimum number of cycles inserted between back-to-back frames transmitted over the FIFO interface. If CRC is appended to frames, 7 cycles (8-bit mode) of non-payload data separates packets.
16	RRX	Enable reset of FIFO receive function. 0 Do not reset the FIFO receiver. 1 Reset the FIFO receiver for as long as this bit is set.
17	RTX	Enable reset of FIFO transmit function. 0 Do not reset the FIFO transmitter. 1 Reset the FIFO transmitter for as long as this bit is set.
18	RXE	Enable FIFO receive function. 0 Disable reception over the FIFO interface, ignoring data presented to the signals. 1 Enable normal reception over the FIFO interface.
19	TXE	Enable FIFO transmit function. 0 Disable transmission over the FIFO interface. 1 Enable normal transmission over the FIFO interface.
20–22	—	Reserved.

**Table 15-105. FIFOCFG Field Descriptions (continued)**

Bits	Name	Description
23	LPB	Loopback enable. 0 Do not loopback data in the FIFO interface. 1 Loopback transmitted data to the FIFO receiver rather than outputting transmitted data to signals.
24	RFC	Enable receive flow control. Setting FFC overrides this bit. 0 Do not allow the FIFO receiver to assert link-level flow control if eTSEC requires it. 1 Allow the FIFO receiver to assert link-level flow control if eTSEC requires it. This is the default setting.
25	TFC	Enable transmit flow control. 0 Do not allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable, resulting in underruns. 1 Allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable and SIGM = 01. This is the default setting.
26	FFC	Force flow control. This can be used by software to stop reception on the FIFO interface. 0 Do not assert link-level flow control via the RXFC signal unless eTSEC requires flow control. 1 Force flow control on the RXFC signal in encoded FIFO packet mode regardless of eTSEC pause requirements.
27	CRCAPP	Append a CRC (CRC-32 algorithm, as per IEEE Std. 802.3) to the end of every transmitted frame. 0 Do not automatically append a CRC to transmitted frames. Allow TxBD[TC], if set, to control when a CRC is appended. 1 Automatically append a CRC to transmitted frames. Ignore TxBD[TC].
28	CRCCHK	Check the CRC (CRC-32 algorithm, as per IEEE Std. 802.3) at the end of every frame. 0 Do not automatically check the last 4 bytes of received frames for a valid CRC. 1 Automatically check the last 4 bytes of received frames for a valid CRC. If a CRC error is detected, or insufficient data is received to recover the CRC, the RxB[CR] bit will be set.
29	—	Reserved
30–31	SIGM	FIFO signaling mode. Determines how the GMII signals are interpreted as framing signals. 00 GMII style mode. 01 Encoded packet mode. 10 Reserved 11 Reserved

### 15.5.3.9 DMA Attribute Registers

This section describes the two eTSEC DMA attribute registers.

#### 15.5.3.9.1 Attribute Register (ATTR)

The attribute register defines memory access attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data. Buffer descriptors may be written with attributes that cause allocation into the L2 cache. Similarly, broad sections of a receive frame header may have attributes attached that cause allocation in the L2 cache. This process of specifying a region of each frame to stash into the L2 cache is referred to as *extraction*, which is specified in conjunction with register ATTRELI. ATTR[ELCWT] only has meaning if ATTRELI[EL] is non-zero. It is important to note that even though portions of received frames may be stashed to L2 cache, this is only a performance optimization as entire frames are still written to off-chip memory regardless of settings in ATTR.

Figure 15-103 describes the definition for the ATTR register.

Offset eTSEC1:0x2\_4BF8; eTSEC3:0x2\_6BF8;

Access: Read/Write

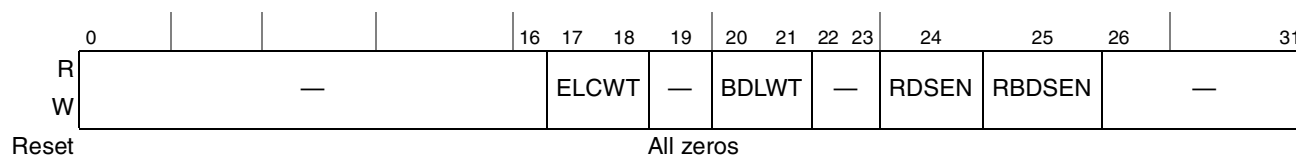


Figure 15-103. ATTR Register Definition

Table 15-106 describes the fields of the ATTR register.

Table 15-106. ATTR Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Specifies the write transaction type to perform for the extracted data. For maximum performance, it is recommended that if ELCWT is set to allocate, BDLWT should also be set to allocate. Writes to cache are always performed with snoop. 00 No allocation performed. 01 Reserved 10 Allocate L2 cache line. 11 Reserved
19	—	Reserved
20–21	BDLWT	Buffer descriptor L2 cache write type. specifies the write transaction type to perform for the buffer descriptor for a receive frame. Writes to cache are always performed with snoop. 00 No allocation performed. 01 Reserved 10 Allocate L2 cache line. 11 Reserved
22–23	—	Reserved
24	RDSSEN	Rx data snoop enable. 0 Disables snooping of all receive frames data to memory unless ELCWT specifies L2 allocation. 1 Enables snooping of all receive frames data to memory.
25	RBDSSEN	RxBD snoop enable. 0 Disables snooping of all receive BD memory accesses unless BDLWT specifies L2 allocation. 1 Enables snooping of all receive BD memory accesses.
26–31	—	Reserved

### 15.5.3.9.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length for extracting received frames. The extract length is typically set to the expected length of extracted packet headers.

Figure 15-104 describes the definition for the ATTRELI register.

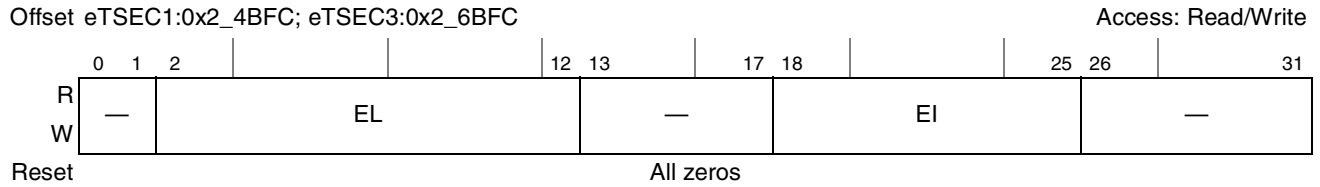


Figure 15-104. ATTRELI Register Definition

Table 15-107 describes the fields of the ATTRELI register.

Table 15-107. ATTRELI Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–12	EL	Extracted length. Specifies the number of bytes, as a multiple of 8 bytes, to extract from the receive frame. The DMA controller uses this field to perform extraction. If set to zero, no extraction is performed.
13–15	—	To ensure that EL is a multiple of 8 bytes, these bits should be written with zero.
16–17	—	Reserved
18–25	EI	Extracted index. Points to the first byte, as a multiple of 64 bytes, within the receive frame from which to begin extracting data.
26–31	—	To ensure that EI is a multiple of 8 bytes, these bits should be written with zero.

### 15.5.3.10 Lossless Flow Control Configuration Registers

When enabled via RCTRL[LFC], the eTSEC will track location of the last free BD in each Rx BD ring via the value of RFBPTR $n$ . Using this pointer and the ring length stored in RQPRM $n$ [LEN], the eTSEC will continuously calculate the number of free BDs in the ring. Whenever the calculated number of free BDs in the ring drops below the pause threshold specified in RQPRM $n$ [FBTHR], the eTSEC will issue link layer flow control. It will continue to assert flow control until the free BD count for each active ring reaches or exceeds RQPRM $n$ [FBTHR]. See Section 15.6.6.1, “Back Pressure Determination via Free Buffers,” for the theory of operation of these registers.

#### 15.5.3.10.1 Receive Queue Parameters 0–7 (RQPRM0–PQPRM7)

The RQPRM $n$  registers specify the minimum number of BDs required to prevent flow control being asserted and the total number of Rx BDs in their respective ring. Whenever the free BD count calculated by the eTSEC for any active ring drops below the value of RQPRM $n$ [FBTHR] for that ring, link level flow control will be asserted. Software must not write to RQPRM $n$  while LFC is enabled and the eTSEC is actively receiving frames. However, software may modify these registers after disabling LFC by clearing RCTRL[LFC]. Note that packets may be lost due to lack of RxBDs while RCTRL[LFC] is clear. Software can prevent packet loss by manually generating pause frames (via TCTRL[TFC\_PAUSE]) to cover the time when RCTRL[LFC] is clear. Figure 15-105 describes the definition for the RQPRM $n$  register.

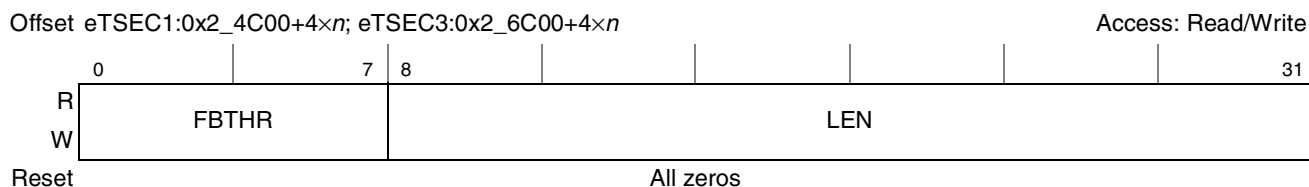


Figure 15-105. RQPRM Register Definition

Table 15-108 describes the fields of the RQPRM register.

Table 15-108. RQPRM Field Descriptions

Bits	Name	Description
0–7	FBTHR	Free BD threshold. Minimum number of BDs required for normal operation. If the eTSEC calculated number of free BDs drops below this threshold, link layer flow control will be asserted.
8–31	LEN	Ring length. Total number of Rx BDs in this ring.

### 15.5.3.10.2 Receive Free Buffer Descriptor Pointer Registers 0–7 (RFBPTR0–RFBPTR7)

The RFBPTR $n$  registers specify the location of the last free buffer descriptor in their respective ring. These registers live in the same 32b address space – and must share the same 4 most significant bits – as RBPTR $n$ . That is, RFBPTR $n$  and its associated RBPTR $n$  must remain in the same 256MB page. Like RBPTR $n$ , whenever RBASE $n$  is updated, RFBPTR $n$  will be initialized to the value of RBASE $n$ . This indicates that the ring is completely empty. As buffers are freed and their respective BDs are returned (by setting the EMPTY bit) to the ring, software is expected to update this register. The eTSEC will then perform modulo arithmetic involving RBASE $n$ , RBPTR $n$  and RFBPTR $n$  to determine the number of free BDs remaining in the ring. If, at any stage, the value written to RFBPTR $n$  matches that of the respective RBPTR $n$  the eTSEC free BD calculation will assume that the ring is now completely empty. For more information on the recommended use of these registers, see Section 15.6.6.1, “Back Pressure Determination via Free Buffers.” Figure 15-106 describes the definition for the RFBPTR $n$  register.

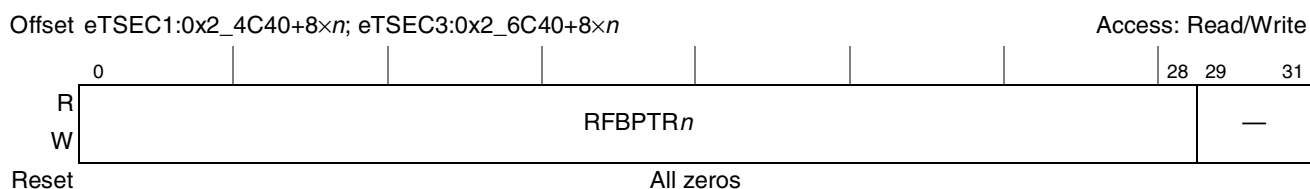


Figure 15-106. RFBPTR0–RFBPTR7 Register Definition

Table 15-109 describes the fields of the RFBPTR $n$  registers.

**Table 15-109. RFBPTR0–RFBPTR7 Field Descriptions**

Bits	Name	Description
0–28	RFBPTR	Pointer to the last free BD in RxBD Ring $n$ . When RBASE $n$ is updated, eTSEC initializes RFBPTR $n$ to the value in the corresponding RBASE $n$ . Software may update this register at any time to inform the eTSEC the location of the last free BD in the ring. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved.

## 15.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

### 15.5.4.1 TBI Transmit Process

The eTSEC's TBI implements the transmit portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. In SerDes mode, packets conveyed across the GMII are encapsulated and encoded into 10-bit symbols and output to the SerDes. In GMII mode, the GMII signals are passed through to the attached GMII PHY.

#### 15.5.4.1.1 Packet Encapsulation

If TX\_EN is de-asserted the eTSEC outputs an idle stream. If TX\_EN is asserted, a Start\_of\_Packet symbol is output. This symbol replaces the first byte of the preamble field. All other bytes of the packet pass through an 8B10B encoding module. After the last byte of the FCS field is signaled via the GMII, the MAC de-asserts TX\_EN. The eTSEC then outputs an End\_of\_Packet symbol. Then, depending on the position of the End\_of\_Packet symbols (being in either an odd or even position) the eTSEC outputs one or two Carrier\_Extend symbols. Following the last Carrier\_Extend symbol, the eTSEC resumes sending idle codes. If, during a packet, the eTSEC wishes to mark a byte invalid, TX\_ER is asserted. The eTSEC, upon detection of TX\_ER, substitutes the data symbol for an Error\_Propagation symbol.

#### 15.5.4.1.2 8B10B Encoding

Every eight-bit data octet has two (not necessarily different) ten-bit symbols associated with it. Depending on the running disparity (the cumulative difference of ones and zeroes) the eTSEC module chooses the appropriate symbol.

Special encapsulation symbols are called ordered\_sets. Ordered\_sets are comprised of one to four ten-bit symbols. Ordered\_sets can be found in Clause 36 of the IEEE 802.3z specification.

#### 15.5.4.1.3 Preamble Shortening

Because the idle ordered\_set comprises two symbols and begins on an even symbols boundary, packets can only begin on an even boundary. However, the GMII has no such restriction and may signal TX\_EN on an odd boundary. If this happens, the eTSEC delays the Start\_of\_Packet symbol, effectively ignoring the first byte of preamble; thus, a seven octet preamble becomes six octets on the Ten-Bit Interface.



## 15.5.4.2 TBI Receive Process

The eTSEC's TBI Implements the receive portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. The Receive portion includes the Synchronization state machine. In SerDes mode, the eTSEC first attempts to acquire synchronization on the link by examining received symbols. Once synchronization is acquired, received packets are decoded and sent across the Receive GMII interface. In GMII mode, the GMII signals are passed through to the MAC.

### 15.5.4.2.1 Synchronization

The eTSEC examines received symbols looking for the seven bit 'comma' string embedded in some special symbols. Both the idle ordered\_set and the Configuration ordered\_set contain a symbol which has the comma. Once a certain number of codes with comma are detected, the eTSEC is considered to have acquired synchronization.

### 15.5.4.2.2 Auto-Negotiation for 1000BASE-X

Once synchronization is acquired, ordered\_sets are decoded. If Configuration ordered\_sets are received, the eTSEC decodes the two octet data field and the sixteen-bit Configuration data is stored and used to Auto-Negotiate with the link partner. In the Receive Configuration Register (RXCR[15:0]) an internal register used to receive all the link partners informations and used to compare to local ability during negotiation. Not visible to user. If, during Auto-Negotiation an invalid symbol is detected, Auto-Negotiation re-starts. After Auto-Negotiation is completed the TBI MII Status Register SR[AN done] in set. In this mode, packets may be received from the link partner.

## 15.5.4.3 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All of the TBI registers are 16 bits wide. The TBI registers are accessed at the offset of the TBI physical address. The eTSEC's TBI physical address is stored in the TBIPA register. Writing to the TBI registers is performed in a way similar to writing to an external PHY, by using the MII management interface. By using TBIPA in place of the PHY address, in the MIIMADD[PHY Address] field, and setting the MIIMADD[Register Address] to the appropriate address offset that corresponds to the register that one wants to read or write (see [Table 15-110](#)), the user can read (set MIIMCOM[read cycle]) or write (writing to MIIMCON[PHY control]) to the TBI block. Refer to the TBI physical address register in [Section 15.5.3.1, "eTSEC General Control and Status Registers,"](#) and the TBI MII register set in [Table 15-110](#). Notice that jitter diagnostics and TBI control are not IEEE 802.3 required registers and are only used for test and control of the eTSEC TBI block. The TBI's TBI control register (TBI) is for configuring the eTSEC ten-bit interface block. However, because this TBI block has an MII management interface (just like any other PHY), it has an IEEE 802.3 register called the control register (CR).

**Table 15-110. TBI MII Register Set**

Offset Address	Name	Access	Size	Section/page
<b>TEN-BIT INTERFACE (TBI) REGISTERS</b>				<a href="#">15.5.4/15-114</a>
0x00	Control (CR)	R/W <sup>1</sup>	16 bits	<a href="#">15.5.4.3.1/15-116</a>

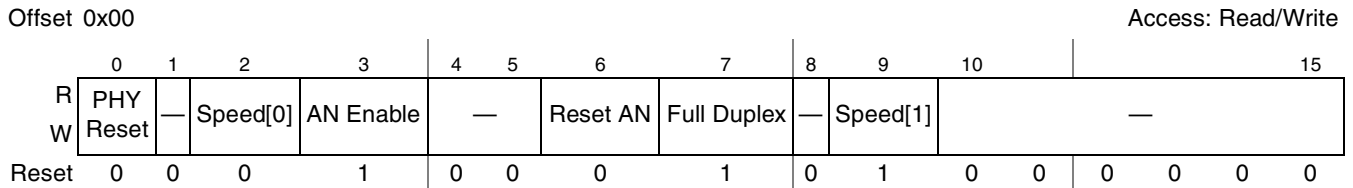
**Table 15-110. TBI MII Register Set (continued)**

Offset Address	Name	Access	Size	Section/page
0x01	Status (SR)	R, LH, LL	16 bits	15.5.4.3.2/15-117
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	15.5.4.3.2/15-117
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	15.5.4.3.4/15-120
0x06	AN expansion (ANEX)	R, LH	16 bits	15.5.4.3.5/15-121
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	15.5.4.3.6/15-122
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	15.5.4.3.7/15-123
0x0F	Extended status (EXST)	R	16 bits	15.5.4.3.8/15-123
0x10	Jitter diagnostics (JD)	R/W	16 bits	15.5.4.3.9/15-124
0x11	TBI control (TBICON)	R/W	16 bits	15.5.4.3.10/15-125

<sup>1</sup> R = Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

### 15.5.4.3.1 Control Register (CR)

Figure 15-107 describes the definition for the CR register.



**Figure 15-107. Control Register Definition**

Table 15-111 describes the fields of the CR register.

**Table 15-111. CR Field Descriptions**

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The internal state of the TBI is reset. This in turn may change the state of the TBI link partner.
1	—	Reserved
2	Speed[0]	Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed. Setting this field controls the speed at which the TBI operates. The table for Speed[1] provides the appropriate encoding. Its default is bit[2] = '0'; bit[9] = '1'.
3	AN Enable	Auto-negotiation enable. This bit is set by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled.
4–5	—	Reserved

Table 15-111. CR Field Descriptions (continued)

Bits	Name	Description															
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation. 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.															
7	Full Duplex	Duplex mode. This bit is set by default. 0 Reserved. 1 Full-duplex operation.															
8	—	Reserved, should be cleared.															
9	Speed[1]	Speed selection. This bit defaults to a set state and should always be set, which corresponds to 1000 Mbps speed. Setting this field controls the speed at which the TBI operates. The following table provides the appropriate encoding. Its default is bit[2] = '0'; bit[9] = '1'. <table border="1" data-bbox="581 659 1149 877" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Maximum Operating Speed</th> <th>Bit 2</th> <th>Bit 9</th> </tr> </thead> <tbody> <tr> <td>Reserved</td> <td>0</td> <td>0</td> </tr> <tr> <td>Reserved</td> <td>1</td> <td>0</td> </tr> <tr> <td>1000 Mbps</td> <td>0</td> <td>1</td> </tr> <tr> <td>Reserved</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Maximum Operating Speed	Bit 2	Bit 9	Reserved	0	0	Reserved	1	0	1000 Mbps	0	1	Reserved	1	1
Maximum Operating Speed	Bit 2	Bit 9															
Reserved	0	0															
Reserved	1	0															
1000 Mbps	0	1															
Reserved	1	1															
10–15	—	Reserved															

### 15.5.4.3.2 Status Register (SR)

Figure 15-108 describes the definition for the SR register.

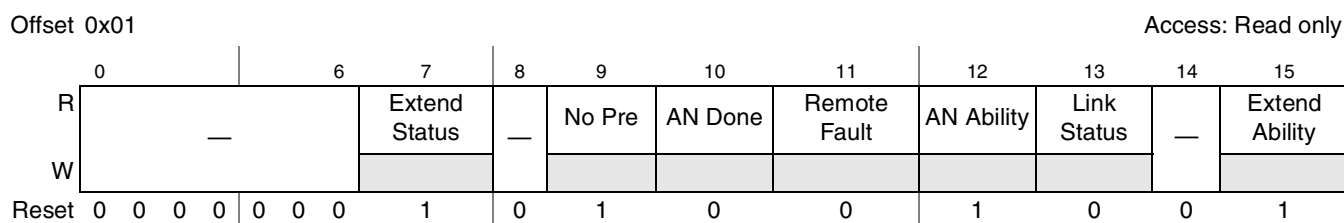


Figure 15-108. Status Register Definition

Table 15-112 describes the fields of the SR register.

Table 15-112. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	Extend Status	This bit indicates that PHY status information is also contained in the Register 15, Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, should be cleared.
9	No Pre	MF preamble suppression enable. This bit indicates whether or not the PHY is capable of handling MII management frames without the 32-bit preamble field. Returns 1, indicating support for suppressed preamble MII management frames. This bit is read-only.

Table 15-112. SR Descriptions

Bits	Name	Description
10	AN Done	Auto-negotiation complete. This bit is read-only and is cleared by default. 0 Either the auto-negotiation process is underway or the auto-negotiation function is disabled. 1 The auto-negotiation process has completed.
11	Remote Fault	Remote fault. This bit is read-only and is cleared by default. Each read of the status register clears this bit. 0 Normal operation. 1 A remote fault condition was detected. This bit latches high in order for software to detect the condition.
12	AN Ability	Auto-negotiation ability. While read as set, this bit indicates that the PHY has the ability to perform auto-negotiation. While read as cleared, this bit indicates the PHY lacks the ability to perform auto-negotiation. Returns 1 on read. This bit is read-only.
13	Link Status	Link status. This bit is read-only and is cleared by default. 0 A valid link is not established. This bit latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, should be cleared.
15	Extend Ability	Extended capability. This bit indicates that the PHY contains the extended set of registers (those beyond control and status). Returns 1 on read. This bit is read-only.

### 15.5.4.3.3 AN Advertisement Register (ANA)

Figure 15-109 describes the definition for the ANA register.

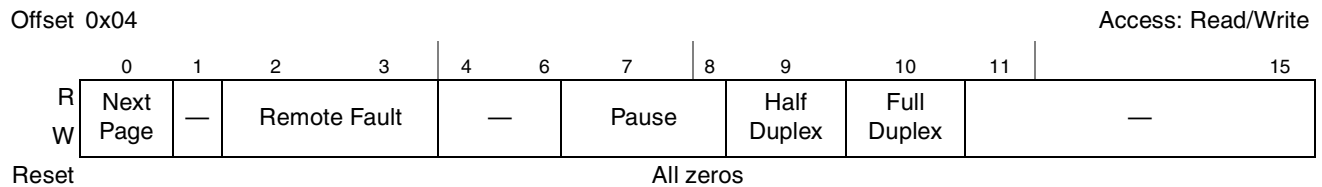


Figure 15-109. AN Advertisement Register Definition

Table 15-113 describes the fields of the ANA register.

Table 15-113. ANA Field Descriptions

Bits	Name	Description
0	Next Page	Next page configuration. The local device sets this bit to either request next page transmission or advertise next page exchange capability. 0 The local device wishes not to engage in next page exchange. 1 The local device has no next pages but wishes to allow reception of next pages. If the local device has no next pages and the link partner wishes to send next pages, the local device shall send null message codes and have the message page set to 0b000_0000_0001, as defined in annex 28C.
1	—	Reserved. (Ignore on read)

Table 15-113. ANA Field Descriptions (continued)

Bits	Name	Description															
2–3	Remote Fault	<p>The local device's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating.</p> <table border="1"> <thead> <tr> <th>RF1 bit[3]</th> <th>RF2 bit[2]</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>No error, link OK</td> </tr> <tr> <td>0</td> <td>1</td> <td>Offline</td> </tr> <tr> <td>1</td> <td>0</td> <td>Link_Failure</td> </tr> <tr> <td>1</td> <td>1</td> <td>Auto-Negotiation_Error</td> </tr> </tbody> </table>	RF1 bit[3]	RF2 bit[2]	Description	0	0	No error, link OK	0	1	Offline	1	0	Link_Failure	1	1	Auto-Negotiation_Error
RF1 bit[3]	RF2 bit[2]	Description															
0	0	No error, link OK															
0	1	Offline															
1	0	Link_Failure															
1	1	Auto-Negotiation_Error															
4–6	—	Reserved, should be cleared.															
7–8	Pause	<p>The local device's PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult <a href="#">Table 15-114</a>.</p> <table border="1"> <thead> <tr> <th>PAUSE bit[8]</th> <th>ASM_DIR bit[7]</th> <th>Capability</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>No PAUSE</td> </tr> <tr> <td>0</td> <td>1</td> <td>Asymmetric PAUSE toward link partner</td> </tr> <tr> <td>1</td> <td>0</td> <td>Symmetric PAUSE</td> </tr> <tr> <td>1</td> <td>1</td> <td>Both symmetric PAUSE and Asymmetric PAUSE toward local device</td> </tr> </tbody> </table>	PAUSE bit[8]	ASM_DIR bit[7]	Capability	0	0	No PAUSE	0	1	Asymmetric PAUSE toward link partner	1	0	Symmetric PAUSE	1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device
PAUSE bit[8]	ASM_DIR bit[7]	Capability															
0	0	No PAUSE															
0	1	Asymmetric PAUSE toward link partner															
1	0	Symmetric PAUSE															
1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device															
9	Half Duplex	<p>Half-duplex capability.            0 Designates local device as not capable of half-duplex operation.            1 Designates local device as capable of half-duplex operation.</p>															
10	Full Duplex	<p>Full-duplex capability.            0 Designates the local device as not capable of full-duplex operation.            1 Designates the local device as capable of full-duplex operation.</p>															
11–15	—	Reserved, should be cleared.															

[Table 15-114](#) describes the resolution of pause priority.

Table 15-114. PAUSE Priority Resolution

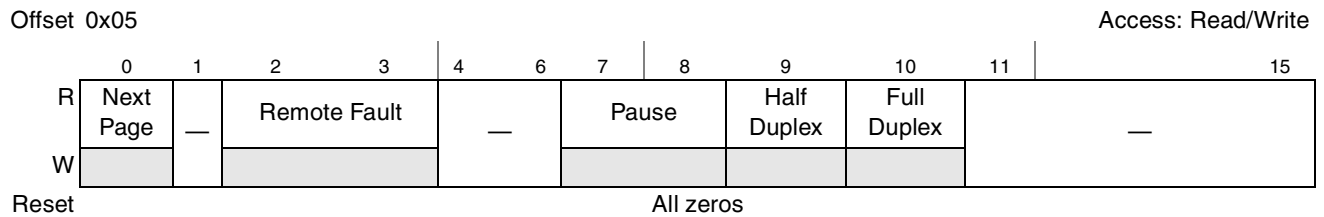
Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	x	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive

**Table 15-114. PAUSE Priority Resolution (continued)**

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	1	1	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	1	Enable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Enable PAUSE receive
1	0	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	0	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive
1	1	0	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	1	0	1	Disable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Disable PAUSE receive
1	1	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive

**15.5.4.3.4 AN Link Partner Base Page Ability Register (ANLPBPA)**

Figure 15-110 describes the definition for the ANLPBPA register.



**Figure 15-110. AN Link Partner Base Page Ability Register Definition**

Table 15-115 describes the fields of the ANLPBPA register.

**Table 15-115. ANLPBPA Field Descriptions**

Bits	Name	Description
0	Next Page	Next page. This bit is read-only. The link partner sets or clears this bit. 0 Link partner has no subsequent next pages or is not capable of receiving next pages. 1 Link partner either requesting next page transmission or indicating the capability to receive next pages.
1	—	Reserved. (Ignore on read)

Table 15-115. ANLPBPA Field Descriptions (continued)

Bits	Name	Description															
2–3	Remote Fault	The link partner's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only. <table border="1" data-bbox="469 373 1253 625"> <thead> <tr> <th>RF1 bit[3]</th> <th>RF2 bit[2]</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>No error, link OK</td> </tr> <tr> <td>0</td> <td>1</td> <td>Offline</td> </tr> <tr> <td>1</td> <td>0</td> <td>Link_Failure</td> </tr> <tr> <td>1</td> <td>1</td> <td>Auto-Negotiation_Error</td> </tr> </tbody> </table>	RF1 bit[3]	RF2 bit[2]	Description	0	0	No error, link OK	0	1	Offline	1	0	Link_Failure	1	1	Auto-Negotiation_Error
RF1 bit[3]	RF2 bit[2]	Description															
0	0	No error, link OK															
0	1	Offline															
1	0	Link_Failure															
1	1	Auto-Negotiation_Error															
4–6	—	Reserved, should be cleared.															
7–8	Pause	Encoding of the link partner's PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult. This bit is read-only <table border="1" data-bbox="363 798 1360 1077"> <thead> <tr> <th>PAUSE bit[8]</th> <th>ASM_DIR bit[7]</th> <th>Capability</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>No PAUSE</td> </tr> <tr> <td>0</td> <td>1</td> <td>Asymmetric PAUSE toward link partner</td> </tr> <tr> <td>1</td> <td>0</td> <td>Symmetric PAUSE</td> </tr> <tr> <td>1</td> <td>1</td> <td>Both symmetric PAUSE and Asymmetric PAUSE toward local device</td> </tr> </tbody> </table>	PAUSE bit[8]	ASM_DIR bit[7]	Capability	0	0	No PAUSE	0	1	Asymmetric PAUSE toward link partner	1	0	Symmetric PAUSE	1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device
PAUSE bit[8]	ASM_DIR bit[7]	Capability															
0	0	No PAUSE															
0	1	Asymmetric PAUSE toward link partner															
1	0	Symmetric PAUSE															
1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device															
9	Half Duplex	Half-duplex capability. This bit is read-only. 0 Link partner is not capable of half-duplex mode. 1 Link partner is capable of half-duplex mode.															
10	Full Duplex	Full-duplex capability. This bit is read-only. 0 Link partner is not capable of full-duplex mode. 1 Link partner is capable of full-duplex mode.															
11–15	—	Reserved, should be cleared.															

### 15.5.4.3.5 AN Expansion Register (ANEX)

Figure 15-111 describes the definition for the ANEX register.

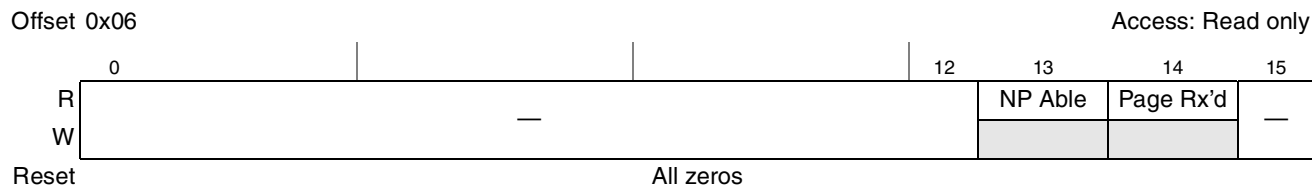


Figure 15-111. AN Expansion Register Definition

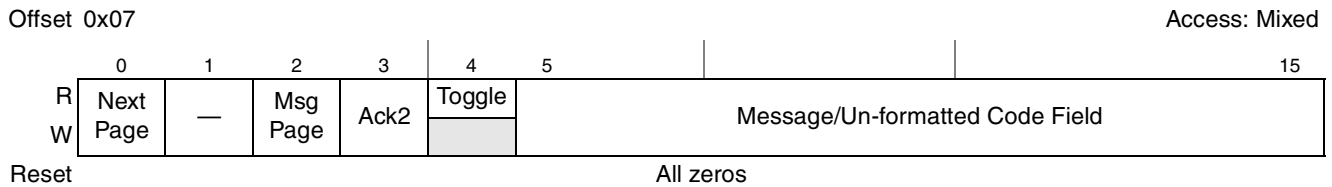
Table 15-116 describes the fields of the ANEX register.

**Table 15-116. ANEX Field Descriptions**

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13	NP Able	Next page able. This bit is read-only and returns 1 on read. While read as set, indicates local device supports next page function.
14	Page Rx'd	Page received. This bit is read-only. The bit clears on a read to the register. 0 Normal operation. 1 A new page was received and stored in the applicable AN link partner ability or AN next page register. This bit latches high in order for software to detect while polling.
15	—	Reserved, should be cleared.

**15.5.4.3.6 AN Next Page Transmit Register (ANNPT)**

Figure 15-112 describes the definition for the ANNPT register.



**Figure 15-112. AN Next Page Transmit Register Definition**

Table 15-117 describes the fields of the ANNPT register.

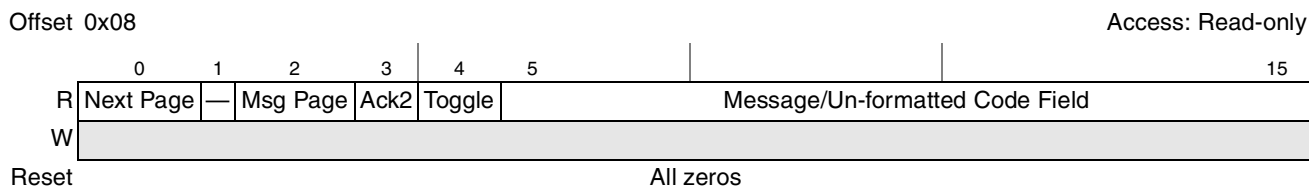
**Table 15-117. ANNPT Field Descriptions**

Bits	Name	Description
0	Next Page	Next page indication. [Reference MII bit 7.15 in IEEE 802.3, 2000 Edition Clause 28.2.4] 0 Last page. 1 Additional next pages to follow.
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. [Reference MII bit 7.13] 0 Unformatted page. 1 Message page.
3	Ack2	Acknowledge 2. Used by the next page function to indicate that the device has the ability to comply with the message. [Reference MII bit 7.12] 0 The local device cannot comply with message. 1 The local device complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. [Reference MII bit 7.11] This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value. [Reference MII field 7.10:0]



### 15.5.4.3.7 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 15-113 describes the definition for the ANLPANP register.



**Figure 15-113. AN Link Partner Ability Next Page Register Definition**

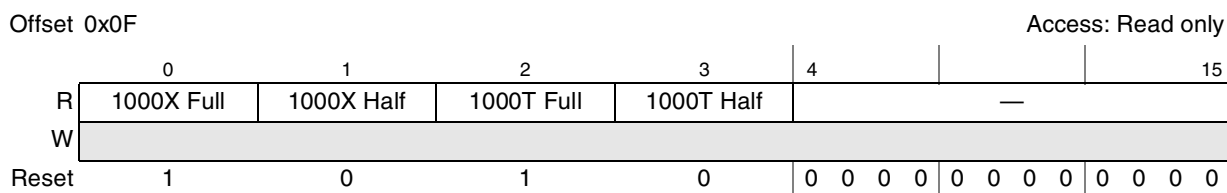
Table 15-118 describes the fields of the ANLPANP register.

**Table 15-118. ANLPANP Field Descriptions**

Bits	Name	Description
0	Next Page	Next page. The link partner sets and clears this bit. 0 Last page from link partner 1 Additional next pages to follow
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. 0 Unformatted page 1 Message page
3	Ack2	Acknowledge 2. Indicates the link partner's ability to comply with the message. 0 Link partner cannot comply with message. 1 Link partner complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/ Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value.

### 15.5.4.3.8 Extended Status Register (EXST)

Figure 15-114 describes the definition for the EXST register.



**Figure 15-114. Extended Status Register Definition**

Table 15-119 describes the fields of the EXST register.

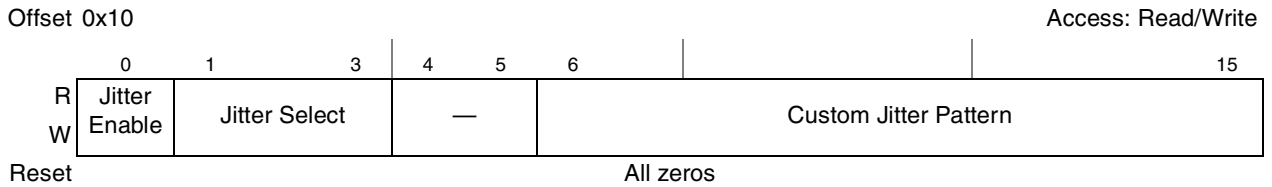
**Table 15-119. EXST Field Descriptions**

Bits	Name	Description
0	1000X Full	1000X full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X full-duplex mode. 1 PHY can operate in 1000BASE-X full-duplex mode.
1	1000X Half	1000X half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X half-duplex mode. 1 PHY can operate in 1000BASE-X half-duplex mode.
2	1000T Full	1000T full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T full-duplex mode. 1 PHY can operate in 1000BASE-T full-duplex mode.
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T half-duplex mode. 1 PHY can operate in 1000BASE-T half-duplex mode.
4–15	—	Reserved

### 15.5.4.3.9 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. In may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns.

Figure 15-115 describes the definition for the JD register.



**Figure 15-115. Jitter Diagnostics Register Definition**

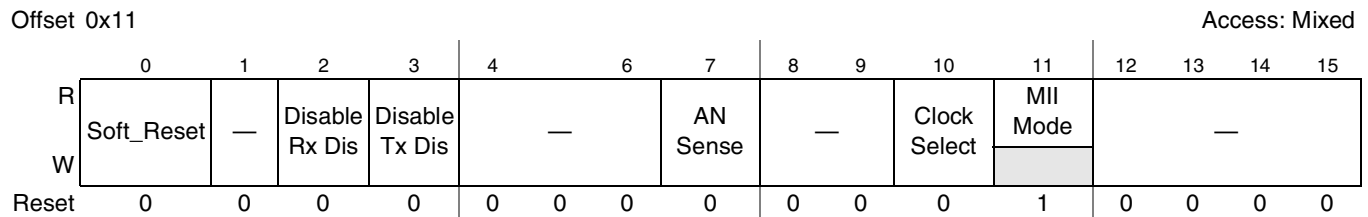
Table 15-120 describes the fields of the JD register.

**Table 15-120. JD Field Descriptions**

Bits	Name	Description																																				
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation. 1 Enable the TBI to transmit the jitter test patterns defined in IEEE 802.3z 36A.																																				
1–3	Jitter Select	<p>Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 00.</p> <table border="1"> <thead> <tr> <th>Jitter Pattern Select</th> <th>bit[1]</th> <th>bit[2]</th> <th>bit[3]</th> </tr> </thead> <tbody> <tr> <td>User defined uses custom jitter pattern, bits 6–15</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>High frequency (+/- D21.5) 10...</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100...</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>Low frequency 1111100000111110000011111000001111100000...</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>Complex pattern (10'h17c,10'h0c9,10'h0e5,10'h2a3, 10'h17c,...)</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>Square Wave (- K28.7) 0011111000001111100000111110000011111000...</td> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>Reserved</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>Reserved</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Jitter Pattern Select	bit[1]	bit[2]	bit[3]	User defined uses custom jitter pattern, bits 6–15	0	0	0	High frequency (+/- D21.5) 10...	0	0	1	Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100...	0	1	0	Low frequency 1111100000111110000011111000001111100000...	0	1	1	Complex pattern (10'h17c,10'h0c9,10'h0e5,10'h2a3, 10'h17c,...)	1	0	0	Square Wave (- K28.7) 0011111000001111100000111110000011111000...	1	0	1	Reserved	1	1	0	Reserved	1	1	1
Jitter Pattern Select	bit[1]	bit[2]	bit[3]																																			
User defined uses custom jitter pattern, bits 6–15	0	0	0																																			
High frequency (+/- D21.5) 10...	0	0	1																																			
Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100...	0	1	0																																			
Low frequency 1111100000111110000011111000001111100000...	0	1	1																																			
Complex pattern (10'h17c,10'h0c9,10'h0e5,10'h2a3, 10'h17c,...)	1	0	0																																			
Square Wave (- K28.7) 0011111000001111100000111110000011111000...	1	0	1																																			
Reserved	1	1	0																																			
Reserved	1	1	1																																			
4–5	—	Reserved																																				
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0x000.																																				

### 15.5.4.3.10 TBI Control Register (TBICON)

Figure 15-116 describes the definition for the TBICON register.



**Figure 15-116. TBI Control Register Definition**

Table 15-121 describes the fields of the TBICON register.

**Table 15-121. TBICON Field Descriptions**

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in the TBI.
1	—	Reserved. (Ignore on read)
2	Disable Rx Dis	Disable receive disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the receive direction.
3	Disable Tx Dis	Disable transmit disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the transmit direction.
4–6	—	Reserved
7	AN Sense	Auto-negotiation sense enable. This bit is cleared by default. 0 IEEE 802.3z Clause 37 behavior is desired, which results in the link not completing. 1 Allow the auto-negotiation function to sense either a Gigabit MAC in auto-negotiation bypass mode or an older Gigabit MAC without auto-negotiation capability. If sensed, auto-negotiation complete becomes true; however, the page received is low, indicating no page was exchanged. Management can then act accordingly.
8–9	—	Reserved
10	Clock Select	Clock select. This bit is cleared by default. 0 Allow the TBI to accept dual split-phase 62.5 MHz receive clocks. 1 Configure the TBI to accept a 125 MHz receive clock from the SerDes/PHY. The 125 MHz clock must be physically connected to 'PMA receive clock 0'.
11	MI Mode	This bit describes the configuration mode of the TBI. The user reads a 1 while the TBI is configured in GMII/MII mode (connected to a GMII/MII PHY) and a 0 while configured in TBI mode (connected to a 1000BASE-X SerDes). Its value is the inverse of ECNTRL[TBIM]. 0 TBI mode. 1 GMII mode.
12–15	—	Reserved

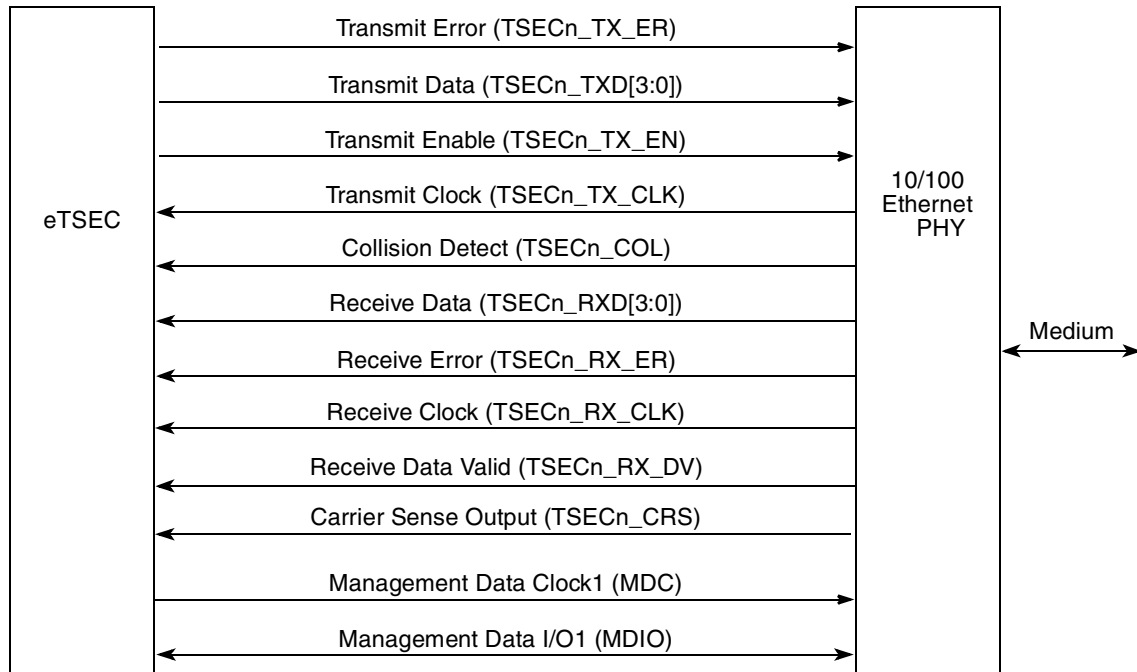
## 15.6 Functional Description

### 15.6.1 Connecting to Physical Interfaces on Ethernet

This section describes how to connect the eTSEC to various interfaces: MII, GMII, RMII, RGMII, TBI, and RTBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification because the PHYs follow the same conventions. (For instance, in the bus TSEC<sub>n</sub>\_TXD[7:0], bit 7 is the msb and bit 0 is the lsb). If a mode does not use all input signals available to a particular eTSEC, those inputs that are not used must be pulled low on the board.

### 15.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the eTSEC. Figure 15-117 depicts the basic components of the MII including the signals required to establish eTSEC module connection with a PHY.



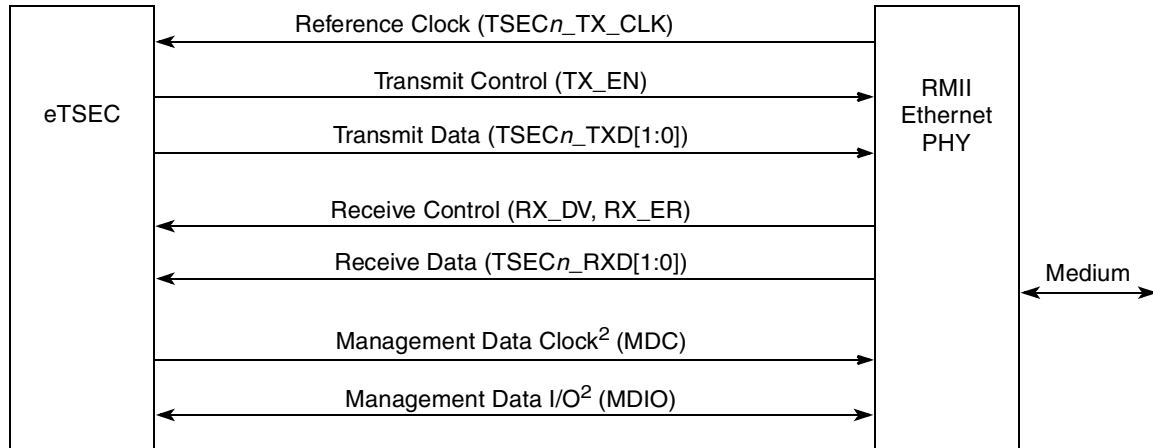
<sup>1</sup> The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

**Figure 15-117. eTSEC-MII Connection**

An MII interface has 18 signals (including the TSECn\_MDC and TSECn\_MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

### 15.6.1.2 Reduced Media-Independent Interface (RMII)

This section describes the reduced media-independent interface (RMII) intended to be used between the PHYs and the GMII MAC. The RMII is a reduced-pin alternative to the IEEE802.3u MII. The RMII reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 18 signals (MII) to 10 signals. To accomplish this objective, the data paths are halved in width and clocked at twice the MII clock frequency, while clocks, carrier sense and error signals have been partly combined. For 100 Mbps operation, the reference clock operates at 50 MHz, whereas for 10 Mbps operation, the clock remains at 50MHz, but only every 10th cycle is used. Figure 15-118 depicts the basic components of the reduced media-independent interface and the signals required to establish an eTSEC's connection with a PHY. The RMII is implemented as defined by the RMII Specification of the RMII Consortium, as of March 20, 1998.

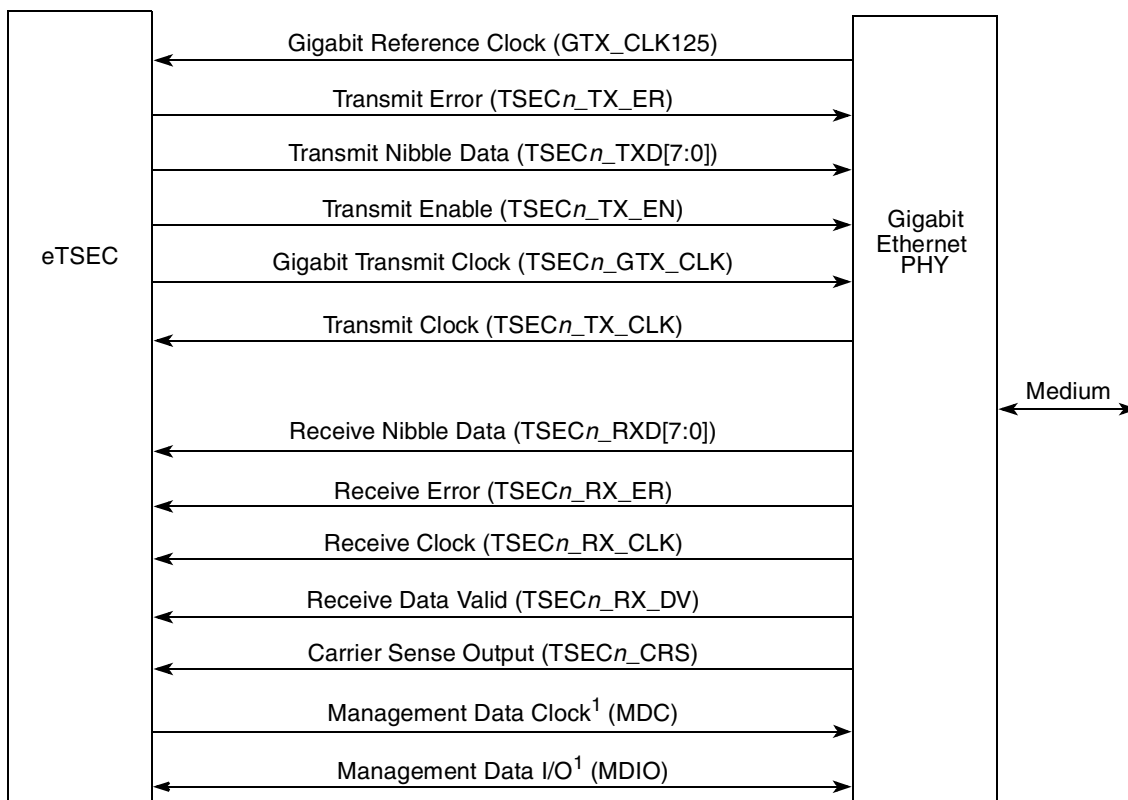


<sup>2</sup> The management signals (MDC and MDIO) are common to all of the Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

**Figure 15-118. eTSEC-RMII Connection**

### 15.6.1.3 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the eTSEC. [Figure 15-119](#) depicts the basic components of the GMII including the signals required to establish the eTSEC module connection with a PHY.



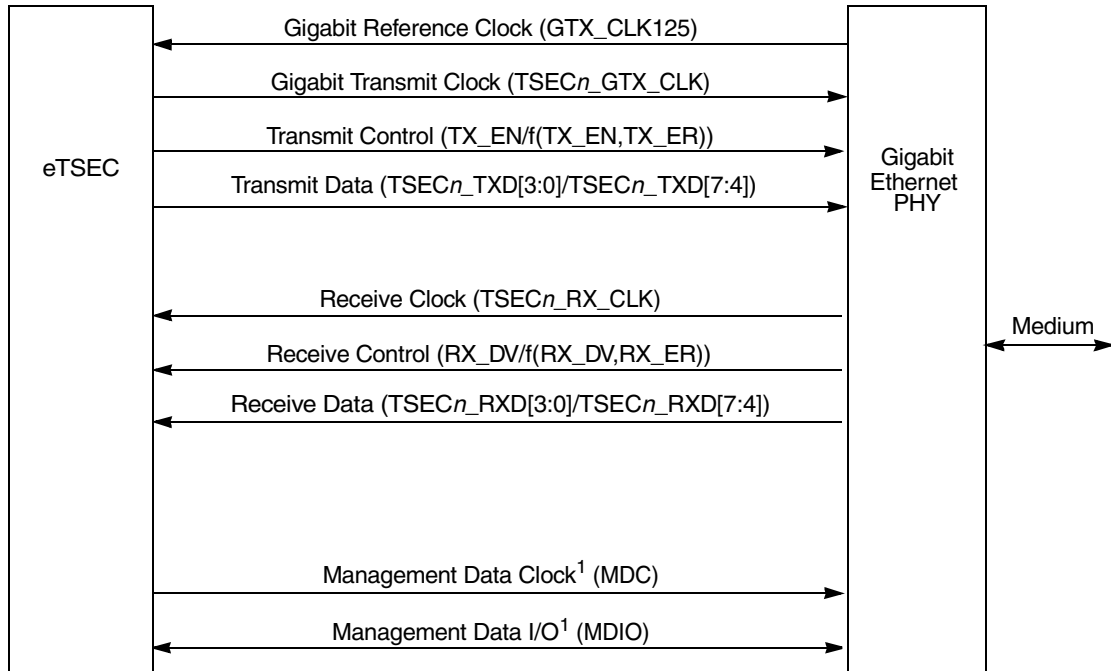
<sup>1</sup> The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

**Figure 15-119. eTSEC-GMII Connection**

A GMII interface has 28 signals (TSEC<sub>n</sub>\_GTX\_CLK + \_GTX\_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

#### 15.6.1.4 Reduced Gigabit Media-Independent Interface (RGMI)

This section describes the reduced gigabit media-independent interface (RGMI) intended to be used between the PHYs and the GMII MAC. The RGMI is an alternative to the IEEE802.3u MII, the IEEE802.3z GMII and the TBI. The RGMI reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 28 signals (GMII) to 15 signals (GTX\_CLK125 included) in a cost effective and technology independent manner. To accomplish this objective, the data paths and all associated control signals are multiplexed using both edges of the clock. For gigabit operation, the clocks operate at 125MHz, and for 10/100 operation, the clocks operate at 2.5 MHz or 25 MHz, respectively. Note that the GTX\_CLK125 input must be provided at 125 MHz for an RGMI interface, regardless of operation speed (1 Gbps, 100 Mbps, or 10 Mbps). [Figure 15-120](#) depicts the basic components of the gigabit reduced media-independent interface and the signals required to establish the gigabit Ethernet controllers' module connection with a PHY. The RGMI is implemented as defined by the RGMI specification Version 1.2a 9/22/00.



<sup>1</sup> The management signals (MDC and MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

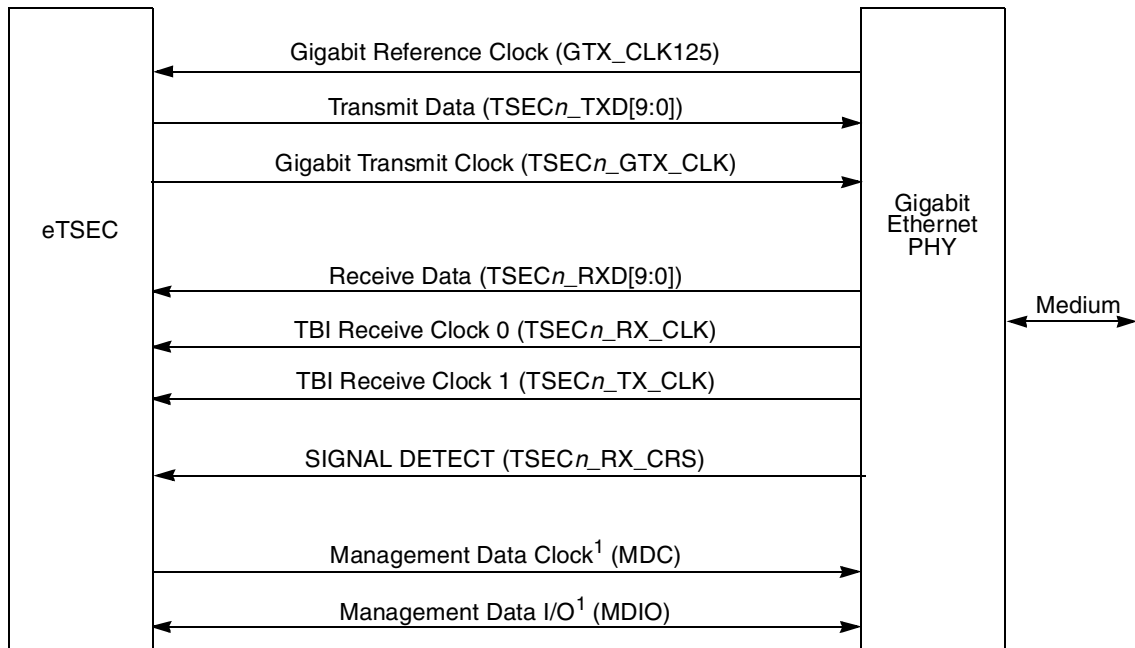
**Figure 15-120. eTSEC-RGMII Connection**



### 15.6.1.5 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the eTSEC to implement a standard SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications.

Figure 15-121 depicts the basic components of the TBI including the signals required to establish eTSEC module connection with a PHY. RBC0 and RBC1 are differential 62.5 MHz receive clocks. If not connected to the TBI PHY, the Signal Detect (SDET) input must be tied high. This causes the eTSEC to begin auto negotiation with the SERDES immediately upon the TBI module being enabled.



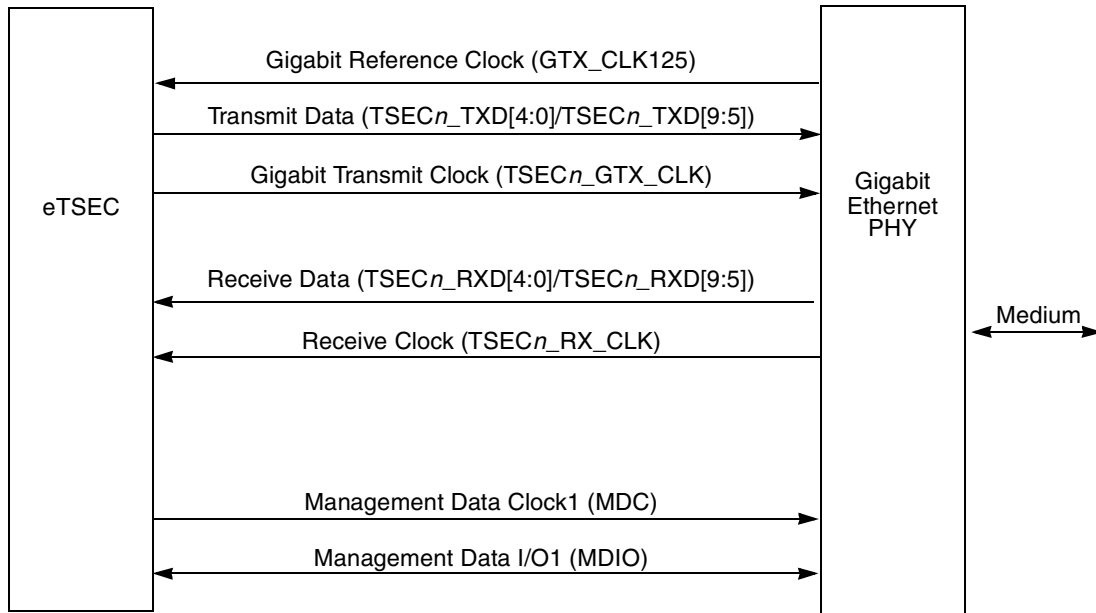
<sup>1</sup> The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

**Figure 15-121. eTSEC-TBI Connection**

A TBI interface has 26 signals (GE\_GTX\_CLK125 included) for connecting to an Ethernet PHY, as defined by IEEE 802.3z GMII and TBI standards.

### 15.6.1.6 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the eTSEC to implement a reduced-pin count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 15-122 depicts the basic components of the RTBI including the signals required to establish eTSEC module connection with a PHY. Note that in RTBI the eTSEC immediately begins auto-negotiation with the SerDes.



<sup>1</sup> The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

**Figure 15-122. eTSEC-RTBI Connection**

A RTBI interface has 15 signals (GE\_GTX\_CLK125 included), as defined by the RGMII specification Version 1.2a 9/22/00, and is intended to be an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI standard for connecting to an Ethernet PHY.

### 15.6.1.7 Ethernet Physical Interfaces Signal Summary

Table 15-122 describes the signal multiplexing for the following interfaces: GMII, MII, TBI, and RMII.

**Table 15-122. GMII, MII, and RMII Signals Multiplexing**

eTSEC Signals			GMII Interface			MII Interface			RMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 25			Frequency [MHz] 50		
Voltage[V] 3.3/2.5			Voltage[V] 3.3			Voltage[V] 3.3			Voltage[V] 3.3		
Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1						
TX_CLK	I	1	TX_CLK	I	1	TX_CLK	I	1	REF_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1	TxD[2]	O	1			
TxD[3]	O	1	TxD[3]	O	1	TxD[3]	O	1			
TxD[4]	O	1	TxD[4]	O	1						
TxD[5]	O	1	TxD[5]	O	1						
TxD[6]	O	1	TxD[6]	O	1						
TxD[7]	O	1	TxD[7]	O	1						
TX_EN	O	1	TX_EN	O	1	TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1	TX_ER	O	1			
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1			
RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1	RxD[2]	I	1			
RxD[3]	I	1	RxD[3]	I	1	RxD[3]	I	1			
RxD[4]	I	1	RxD[4]	I	1						
RxD[5]	I	1	RxD[5]	I	1						
RxD[6]	I	1	RxD[6]	I	1						
RxD[7]	I	1	RxD[7]	I	1						
RX_DV	I	1	RX_DV	I	1	RX_DV	I	1	CRS_DV	I	1
RX_ER	I	1	RX_ER	I	1	RX_ER	I	1	RX_ER	I	1
COL	I	1				COL	I	1			
CRS	I	1				CRS	I	1			
<b>Sum</b>		25	<b>Sum</b>		23	<b>Sum</b>		16	<b>Sum</b>		8

Table 15-124 describes the signal multiplexing for RGMII, TBI, and RTBI interfaces.

**Table 15-123. RGMII, TBI, and RTBI Signals Multiplexing**

eTSEC Signals			RGMII Interface			TBI Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 3.3			Voltage[V] 2.5		
Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1				RX_CLK1	I	1			
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1				TCG[4]	O	1			
TxD[5]	O	1				TCG[5]	O	1			
TxD[6]	O	1				TCG[6]	O	1			
TxD[7]	O	1				TCG[7]	O	1			
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[8]	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1				TCG[9]	O	1			
RX_CLK	I	1	RX_CLK	I	1	RX_CLK0	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1				RCG[4]	I	1			
RxD[5]	I	1				RCG[5]	I	1			
RxD[6]	I	1				RCG[6]	I	1			
RxD[7]	I	1				RCG[7]	I	1			
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[8]	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1				RCG[9]	I	1			
COL	I	1									
CRS	I	1				SDET	I	1		I	
<b>Sum</b>		25	<b>Sum</b>		12	<b>Sum</b>		24	<b>Sum</b>		12

Table 15-124. RGMII and RTBI Signals Multiplexing

eTSEC Signals			RGMII Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 2.5		
Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals	Signals (TSEC <sub>n</sub> )	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1						
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1						
TxD[5]	O	1						
TxD[6]	O	1						
TxD[7]	O	1						
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1						
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1						
RxD[5]	I	1						
RxD[6]	I	1						
RxD[7]	I	1						
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1						
COL	I	1						
CRS	I	1					I	
<b>Sum</b>		25	<b>Sum</b>		12	<b>Sum</b>		12

Table 15-125. RGMII Signals Multiplexing

eTSEC Signals			RGMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals (TSECn <sub>-</sub> )	I/O	No. of Signals	Signals (TSECn <sub>-</sub> )	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1			
TxD[5]	O	1			
TxD[6]	O	1			
TxD[7]	O	1			
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1			
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1			
RxD[5]	I	1			
RxD[6]	I	1			
RxD[7]	I	1			
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1			
COL	I	1			
CRS	I	1			
<b>Sum</b>		25	<b>Sum</b>		12

Table 15-126 describes the signals shared by all interfaces.

**Table 15-126. Shared Signals**

Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	Management interface clock
GTX_CLK125	I	1	Reference clock
<b>Sum</b>			

## 15.6.2 Connecting to FIFO Interfaces

This section describes how to connect an eTSEC to third-party communication devices, including users' ASICs and FPGAs, via the FIFO interface.

Each eTSEC provides an 8-bit full-duplex packet FIFO interface port that bypasses the Ethernet MAC, but re-uses the GMII signals. As a result, the FIFO interface normally does not impose the overheads of Ethernet framing. The FIFO interface operates synchronously, at a maximum frequency defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and 3.2:1 (platform:TxClk) in encoded mode providing OC-48 full-duplex transfer rates. For example, a FIFO frequency of 127 MHz in GMII mode requires a platform frequency of 533 MHz; a FIFO frequency of 200 MHz in encoded mode requires a platform frequency of 667 MHz; a FIFO frequency of 167 MHz in encoded mode requires a platform frequency of 533 MHz.

Bare IP packets—with an optional 32-bit CRC check sequence—can be transferred to the eTSEC directly. The eTSEC Tx and Rx FIFOs, TOE functions, and DMA continue to be used in packet FIFO mode.

The ECNTL[FIFM] bit determines whether eTSEC is communicating with its Ethernet MAC or FIFO interface.

- 8-bit packet FIFO
  - The GMII signals of each eTSEC can be used to create a FIFO port, therefore eTSEC can support up to simultaneous 8-bit FIFO interfaces. Choosing between 8-bit FIFO and Ethernet affects each eTSEC independently, therefore a mix of FIFO and Ethernet interfaces can be configured.
  - The data signals of GMII and 8-bit FIFO remain the same. The data valid (RX\_DV, TX\_EN) and error (RX\_ER, TX\_ER) signals are used to signal framing information. If required, the collision (COL) and carrier sense (CRS) signals can be used in an encoded mode to provide link-level flow control.

The possible port mode combinations supported across both eTSECs are shown in [Table 15-127](#).

The following restrictions apply in any of the FIFO modes:

**Table 15-127. Port Mode Combinations Supported by eTSECs**

Mode Option	eTSEC1 Signals	eTSEC3 Signals
Ethernet only	All Ethernets	All Ethernets
8-bit FIFO Only	8-bit FIFO port 1	8-bit FIFO port 2
8-bit FIFO & Ethernet, mixed	8-bit FIFO port 1 or Reduced Ethernets	8-bit FIFO port 2 or Reduced Ethernets

- Transferred packets must be a minimum of 10 bytes, and no more than 9600 bytes in length.
- Although TCP/IP offload is supported, the receive queue filter table must be limited to as many entries as eTSEC can search every packet. See [Section 15.6.5.1.1, “Filing Rules,”](#) for guidance on how to determine maximum table size for an application.
- eTSEC requires received packets to have a minimum inter-packet gap of three cycles.
- On transmission, the minimum inter-packet gap is three cycles if CRC is not automatically appended; each CRC data beat reduces this minimum inter-packet gap down to no less than one cycle.

No Ethernet-specific features (such as MAC address matching) or layer 2 properties (such as Ethertype) are available in FIFO mode.

### 15.6.2.1 Flow Control

In the encoded (non GMII-style) FIFO modes, link-level flow control is provided to the eTSEC transmitter on the COL signal of the controlling eTSEC, while back pressure to the remote transmitter is sent on the CRS signal (which acts as an output signal only in FIFO mode). Owing to the synchronization delay of responding to flow control on signal COL, the eTSEC cannot stop transmission immediately, but may require up to 8 clock cycles before transmission is paused. The eTSEC will issue flow control either when software forces it (via the FIFOCFG[FFC] bit), or when the Rx FIFO reaches its high watermark.

### 15.6.2.2 CRC Appending and Checking

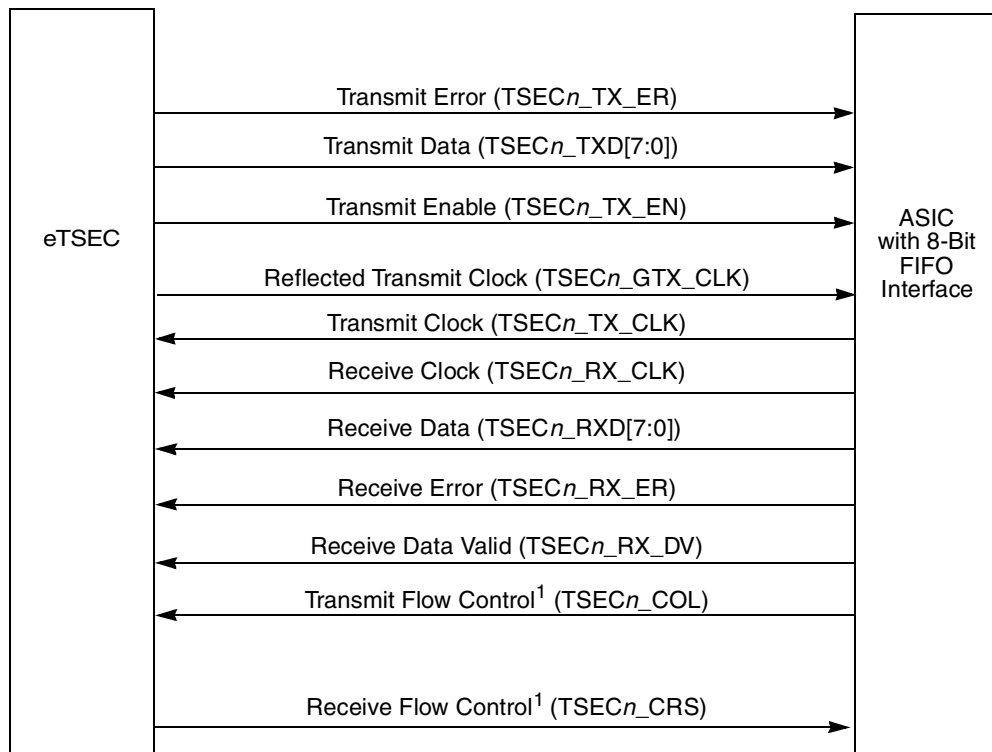
If FIFOCFG[CRCAPP] is enabled, the FIFO interface will automatically append a 4-byte CRC to each transmitted packet. Alternatively, if FIFOCFG[CRCAPP] is cleared, TxBD[TC] provides a per-packet override to append CRC. The IEEE 802.3 standard CRC-32 algorithm is used, where the least significant bit of each byte (TXD[0]) is combined into the CRC ahead of the most significant bit (TXD[7]). Accordingly, the CRC result, CRC[31:0] is transmitted onto the interface in bit-reversed order, CRC[24:31], CRC[16:23], CRC[8:15], CRC[0:7].

Automatic checking of CRC-32 checksums received over the FIFO interface is enabled by setting FIFOCFG[CRCCHK]. CRC errors are recorded in the RxB[CR] flag of every last buffer. Like transmit, the receiver combines data into the CRC in the order least significant data bit (RXD[0]) to most significant bit (RXD[7]). The last 4 bytes of the packet are assumed to be CRC whenever FIFOCFG[CRCCHK] is enabled, and these bytes are returned as part of the data buffer.



### 15.6.2.3 8-Bit GMII-Style Packet FIFO Mode

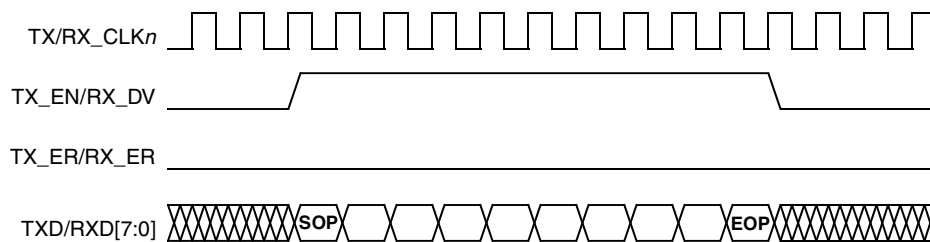
Figure 15-123 depicts the signals required to establish eTSEC module connection with an external device using the 8-bit FIFO interface.



<sup>1</sup>The flow control signals (TSEC<sub>n</sub>\_CRS and TSEC<sub>n</sub>\_COL) are common to all of the FIFO modes. TSEC<sub>n</sub>\_CRS becomes an output signal in FIFO modes only.

**Figure 15-123. eTSEC-FIFO (8-Bit) Connection**

The 8-bit FIFO interface has 25 signals (including the flow control signals). Illustrative timing of the GMII-style FIFO mode is shown in Figure 15-124.



**Figure 15-124. 8-Bit GMII-Style Packet FIFO Timing**

The encoding of the eTSEC GMII signals in this FIFO mode is shown in [Table 15-128](#).

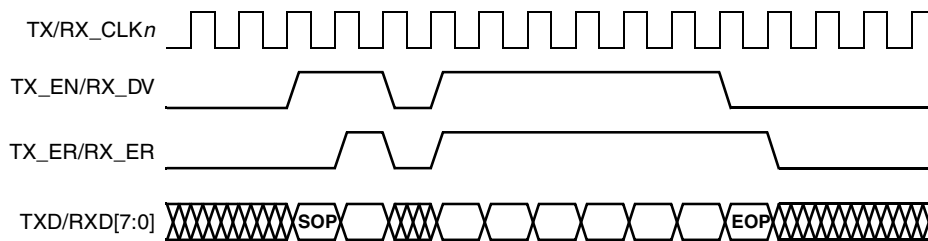
**Table 15-128. Signal Encoding for GMII-Style 8-Bit FIFO**

Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	0 to 1 transition at start of cycle	0
Valid data	1	0
Valid data, end of packet	1 to 0 transition at end of cycle	0
Error	1	1 until TX_EN/RX_DV falls

In this mode flow control can control only the decision to continue transmitting packets, as packet transfers cannot be suspended once started.

#### 15.6.2.4 8-Bit Encoded Packet FIFO Mode

The encoded packet 8-bit FIFO mode uses the signals shown in [Figure 15-125](#). The control lines encode four states that can be associated with each beat of data. This mode should be used where invalid bytes can appear between the start and end of packet. Illustrative timing of the encoded packet FIFO mode is shown in [Figure 15-125](#).



**Figure 15-125. 8-Bit Encoded Packet FIFO Timing**

The encoding of the eTSEC GMII signals in this FIFO mode is shown in [Table 15-129](#).

**Table 15-129. Signal Encoding for Encoded 8-Bit FIFO**

Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	1	0
Valid data	1	1
Valid data, end of packet	0	1
Data not valid	0	0

In this mode flow control can cause an indefinite number of invalid data bytes to be transferred. This is the only mode in which an empty eTSEC Tx FIFO also causes a string of invalid data bytes to be transmitted rather than causing an underrun error.

#### 15.6.2.5 FIFO Interface Signal Summary

Refer to [Section 15.7.1.7, “8-Bit FIFO Mode”](#) for interface signal details.

## 15.6.3 Gigabit Ethernet Controller Channel Operation

This section describes the operation of the eTSEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Frame filtering and receive filing algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

### 15.6.3.1 Initialization Sequence

This sections describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the eTSEC.

#### 15.6.3.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All eTSEC's registers and control logic are reset to their default states after a hard reset has occurred. In this state, each eTSEC behaves like a PowerQUICC III device, except for the absence of out-of-sequence TxBD features. That is, initially TCP/IP off-load is disabled and only single RxBD and TxBD rings are accessible.

#### 15.6.3.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic eTSEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. See [Table 15-3](#) for the register list. [Table 15-130](#) describes the minimum steps for register initialization.

**Table 15-130. Steps for Minimum Register Initialization**

Description
1. Set and clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII Mgmt Interface
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT
7. Initialize IMASK
8. Initialize RCTRL
9. Initialize DMACTRL

After the initialization of registers is performed, the user must execute the following steps in the order described below to bring the eTSEC into a functional state (out of reset):

1. Write to the MACCFG1 register and set the appropriate bits. These need to include RX\_EN and TX\_EN. To enable flow control, Rx\_Flow and Tx\_Flow should also be set.

2. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE $n$  registers. A minimum of two buffer descriptors per ring is required, unless the ring is disabled. Setting the ring to a size of one causes the same frame to be transmitted twice. If TCP/IP off-load is to be enabled, the TxBD[TOE] bit must be set for each frame.
3. Likewise, for the reception of Ethernet frames, the receive queue (or queues) must be ready, with its RxBD pointed to by the RBASE $n$  registers. If TCP/IP off-load is to be enabled, RCTRL[PRSDEP] must be set to the required off-load level. Both transmit and receive can be gracefully stopped after transmission and reception begins.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. The DMACTRL[GRS] must be cleared if the receiver had been previously stopped. Refer to the DMACTRL register section, and [Section 15.6.7.1, “Data Buffer Descriptors,”](#) for more information.

### 15.6.3.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft-reset to and/or reconfiguring the MAC with new parameters, the user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors will be used, the user must load the pointers into the TBASE registers. Likewise if a new set of Rx buffer descriptors will be used, the RBASE registers must be written with new pointers.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GRS/GTS bits in DMACTRL register
2. Poll GRSC/GTSC bits in IEVENT register until both are set
3. Set SOFT\_RESET bit in MACCFG1 register (Note that SOFT\_RESET must remain set for at least 3 TX clocks before proceeding.)
4. Clear SOFT\_RESET bit in MACCFG1 register
5. Load TDBPH, TBASEH, TBASE0–TBASE7 with new Tx BD pointers
6. Load RDBPH, RBASEH, RBASE0–RBASE7 with new Rx BD pointers
7. Setup other MAC registers (MACCFG2, MAXFRM, etc.)
8. Setup group address hash table (GADDR0–GADDR15) if address filtering is required
9. Setup receive frame filter table (via RQFAR, RQFCR, and RQFPR) if filtering to multiple RxBD rings is required
10. Setup WWR, WOP, TOD bits in DMACTRL register
11. Enable transmit queues in TQUEUE, and ensure that the transmit scheduling mode is correctly set in TCTRL.
12. Enable receive queues in RQUEUE, and optionally set TOE functionality in RCTRL.
13. Clear THLT and TXF bits in TSTAT register by writing 1 to them

14. Clear QHLT and RXF bits in RSTAT register by writing 1 to them.
15. Clear GRS/GTS bits in DMACTRL (do not change other bits)
16. Enable Tx\_EN/Rx\_EN in MACCFG1 register

### 15.6.3.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the eTSEC begins to poll the first transmit buffer descriptor (TxBD) in TxBD ring 0 every 512 transmit clocks. If TxBD[R] is set, and the TxBD ring is scheduled for transmission, the eTSEC begins copying the associated transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the GMII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, setting the DMACTRL[TOD] eliminates waiting for the next poll and a DMA transfer of the transmit data buffers can begin immediately. The transmission begins once all data for the frame is loaded into the Tx FIFO or sufficient transmit data (determined by the Tx FIFO threshold register) is in the Tx FIFO. If the line is not busy, the MAC transmit logic asserts TX\_EN and sends the 7-octet preamble sequence, 1-octet start of frame delimiter, and frame information in that order. If the line is busy, the controller waits for the carrier sense signal, CRS, to remain inactive for 60 bit times (60 clocks) and transmission begins after an additional 36 bit times (96 bit times after CRS became active). In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap (96 bit times) regardless of CRS.

In half-duplex mode (MACCFG2[Full Duplex] is cleared) the MAC defers transmission if the line is busy (CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after CRS originally became negated). If CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is re-transmitted in case of a collision. This avoids unnecessary memory traffic.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap must be enforced.

The transmitter implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter's request to send data until the pause duration is over. If the MAC is currently sending data after a pause frame has been received and processed, the MAC finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. In addition, the transmitter supports transmission of flow control frames via TCTRL[TFC\_PAUSE]. The transmit pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD

- MACCFG2[*PAD/CRC*] is set
- MACCFG2[*CRC*] is set

The Tx\_EN is negated after the FCS is sent. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[*PAD/CRC*] is set, the Ethernet controller pads any frame shorter than 64 bytes with zero bytes to make up the minimum length.

To pause transmission, or rearrange the transmit queue, set DMACTRL[*GTS*]. This can be useful for transmitting expedited data ahead of previously-linked buffers or for error situations. If this bit is set, the eTSEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until all queued frames in the Tx FIFO have been disposed of. The IEVENT[*GTSC*] interrupt occurs once the graceful transmit stop operation is completed. After the DMACTRL[*GTS*] is cleared, the eTSEC resumes transmission with the next frame.

While the eTSEC is in 10/100Mbps mode it sends bytes least-significant nibble first and each nibble is sent lsb first. While it is in 1000Mbps mode it sends bytes LSB first.

### 15.6.3.4 Gigabit Ethernet Frame Reception

The eTSEC Ethernet receiver is designed to work with little core intervention and can perform data extraction, address recognition, CRC checking, short frame checking, and maximum frame-length checking.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[*RX\_EN*]. The Ethernet receiver is enabled and immediately starts processing receive frames. The MAC checks for when TSECn\_RX\_DV is asserted and as long as TSECn\_COL remains negated (full-duplex mode ignores TSECn\_COL), the MAC looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped (unless MACCFG2[*PreAM RxEN*] is set) and the frame begins to be processed. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the eTSEC controller begins to perform the frame recognition function through destination address (DA) recognition (see [Section 15.6.3.7, “Frame Recognition”](#)). Based on this match the frame can be accepted or rejected. The receiver can filter frames based on individual (unicast), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, system bus usage is not wasted on frames unwanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from either queue 0 or the queue determined by the filter. If the RxBD is not being used by software (RxBD[*E*] is set), the eTSEC starts transferring the incoming frame. RxBD[*F*] is set for the first RxBD used for any particular receive frame. If the current RxBD is not available for the received frame, a receive busy error condition is raised in IEVENT[*BSY*].

After the buffer is filled, the eTSEC clears RxBD[E] and, if RxBD[I] is set, generates an interrupt. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table. If it is empty, the controller continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicate LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBLR). The smallest valid value is 64 bytes, with larger values being some integral multiple of 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize an oversized frame condition.

Receive frames are not truncated when they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the eTSEC receiver performs a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signaled after the graceful receive stop operation is completed. While in this mode the user can write to registers that are accessible to both the user and the eTSEC hardware without fear of conflict, and finally clear IEVENT[GRSC]. After DMACTRL[GRS] is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

### 15.6.3.5 Ethernet Preamble Customization

By default eTSEC generates a standard Ethernet preamble sequence prior to transmitting frames. However, the user can substitute a custom preamble sequence for the purpose of controlling switching equipment at the receiver, particularly at 100/1000Mbps speeds. In FIFO mode preamble customization is ignored; in any RMII mode only the standard preamble can be transmitted.

eTSEC normally searches for and discards the standard Ethernet preamble sequence upon receiving frames. Part of the received preamble sequence can be optionally recovered and returned as part of the frame data, making it visible to user software. Note however, that no preamble is received in FIFO mode, and preamble cannot be recovered in any RMII mode. Note that it is also possible for the first two bytes of custom preamble (PreOct0 and PreOct1) to be lost in during conversion to ten-bit code groups in the PCS sub-layer. Thus it is recommended that any custom preamble start at PreOct2.

### 15.6.3.5.1 User-Defined Preamble Transmission

To substitute a custom preamble, the user must ensure that:

- MACCFG2[PreAm TxEN] bit is set
- The first TxBD of every frame containing a custom preamble has its PRE bit set
- An 8-byte custom preamble sequence appears before the Ethernet DA field in the first transmit data buffer

The definition of the 8-byte custom preamble sequence is shown in [Figure 15-126](#).

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0							PreOct1								
2–3	PreOct2							PreOct3								
4–5	PreOct4							PreOct5								
6–7	PreOct6															

**Figure 15-126. Definition of Custom Preamble Sequence**

The fields of the custom preamble sequence are described in [Table 15-131](#). It should be noted that use of preamble octets matching the standard start of frame delimiter (0xD5) can be expected to trigger premature frame reception by the receiving station.

**Table 15-131. Custom Preamble Field Descriptions**

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of custom transmit preamble. This is the first octet of preamble sent.
	8–15	PreOct1	Octet #1 of custom transmit preamble. This is the second octet of preamble sent.
2–3	0–7	PreOct2	Octet #2 of custom transmit preamble. This is the third octet of preamble sent.
	8–15	PreOct3	Octet #3 of custom transmit preamble. This is the fourth octet of preamble sent.
4–5	0–7	PreOct4	Octet #4 of custom transmit preamble. This is the fifth octet of preamble sent.
	8–15	PreOct5	Octet #5 of custom transmit preamble. This is the sixth octet of preamble sent.
6–7	0–7	PreOct6	Octet #6 of custom transmit preamble. This is the seventh octet of preamble sent. The last octet (the start of frame delimiter) is generated by the MAC automatically.
	8–15	—	Reserved; should be cleared.

### 15.6.3.5.2 User-Visible Preamble Reception

To return the received preamble, the user must ensure that:

- MACCFG2[PreAm RxEN] bit is set
- Space for an 8-byte preamble sequence is allowed before the Ethernet DA field in the first receive data buffer of each frame



The definition of the 8-byte received preamble sequence is shown in [Figure 15-127](#).

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0							PreOct1								
2–3	PreOct2							PreOct3								
4–5	PreOct4							PreOct5								
6–7	PreOct6															

**Figure 15-127. Definition of Received Preamble Sequence**

The fields of the received preamble sequence are described in [Table 15-132](#). Should the received preamble be shorter than the 7-octet sequence defined by IEEE Std. 802.3, initial bytes of the received preamble sequence hold undefined values. The standard start of frame delimiter (0xD5) is always omitted. Note that preamble extraction is not possible in RMII mode.

**Table 15-132. Received Preamble Field Descriptions**

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of received preamble. This is the first octet of preamble received.
	8–15	PreOct1	Octet #1 of received preamble. This is the second octet of preamble received.
2–3	0–7	PreOct2	Octet #2 of received preamble. This is the third octet of preamble received.
	8–15	PreOct3	Octet #3 of received preamble. This is the fourth octet of preamble received.
4–5	0–7	PreOct4	Octet #4 of received preamble. This is the fifth octet of preamble received.
	8–15	PreOct5	Octet #5 of received preamble. This is the sixth octet of preamble received.
6–7	0–7	PreOct6	Octet #6 of received preamble. This is the seventh octet of preamble received. The last octet (the start of frame delimiter) is discarded.
	8–15	—	Reserved

### 15.6.3.6 RMON Support

By using promiscuous mode, the eTSEC can automatically gather network statistics required for remote network interface monitoring. The RMON MIB group 1, RMON MIB group 2, RMON MIB group 3, RMON MIB group 9, RMON MIB2, and the 802.3 Ethernet MIB are supported. For RMON statistics and their corresponding counters, see the memory map.

### 15.6.3.7 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

#### 15.6.3.7.1 Destination Address Recognition and Frame Filtering

The eTSEC can perform layer 2 frame filtering on the basis of destination Ethernet address (DA), as illustrated by the flowchart in [Figure 15-128](#).

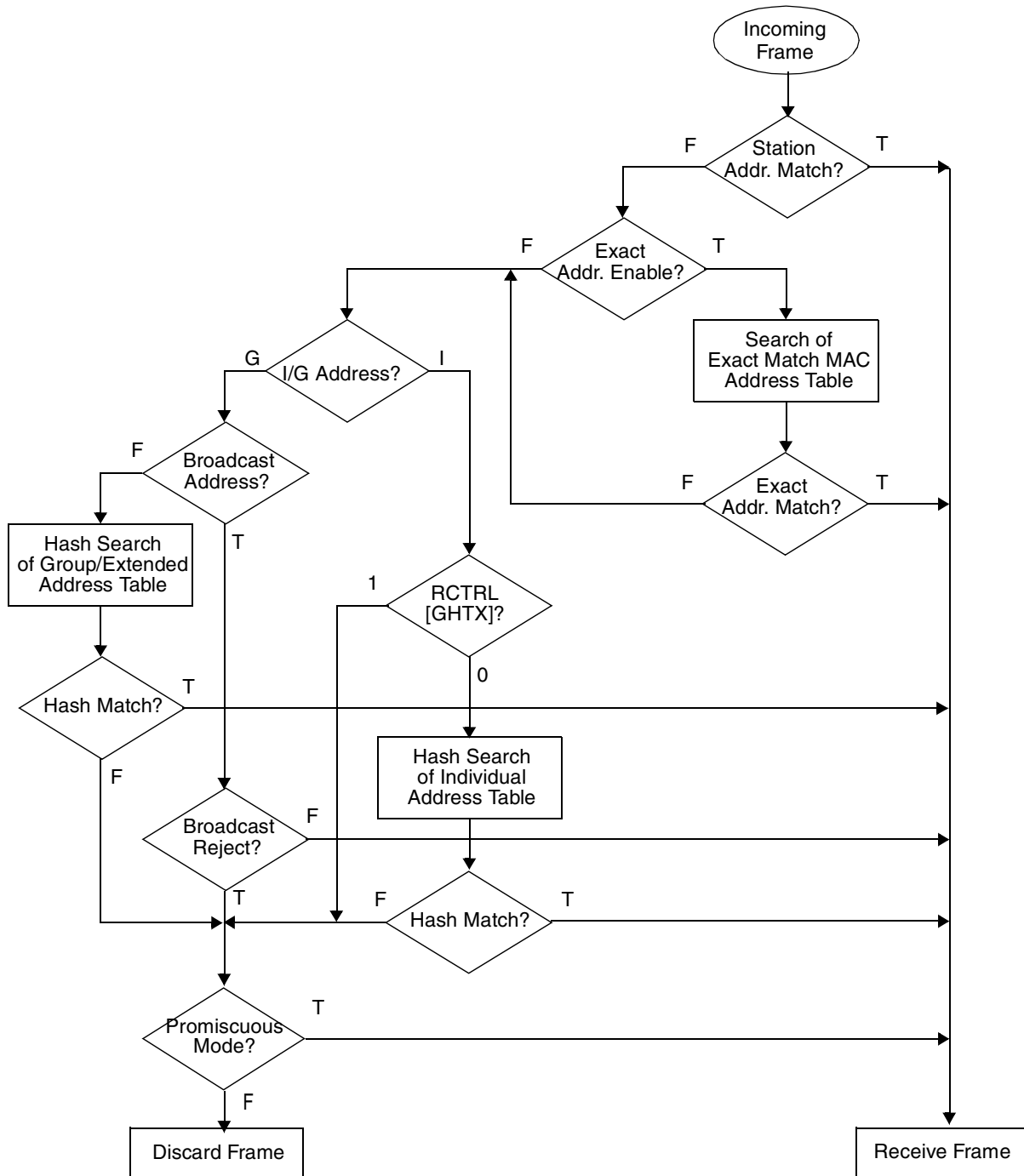


Figure 15-128. Ethernet Address Recognition Flowchart

In promiscuous mode, the eTSEC accepts all received frames regardless of DA. Note, however, that Ethernet frame filtering simply restricts the traffic seen by the receive queue filter. Therefore even in promiscuous mode it remains possible to program the filter to reject frames based on their higher-layer header contents.

In the case of an individual address, the DA field of the received frame is compared with the physical address that the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, and exact MAC address matching is enabled via RCTRL[EMEN], the controller performs address recognition on the multiple MAC addresses written to the MACxADDR1 and MACxADDR2 registers. These virtual addresses give a particular eTSEC the ability to mirror other MACs on the network, which caters for router redundancy protocols, such as HSRP and VRRP.

If exact MAC address matching is not enabled, the eTSEC determines whether DA is a group or individual address. If DA is the standard broadcast address, and broadcast addresses are not rejected, the frame is accepted. If any other group address is received, the eTSEC looks-up the DA by means of the group hash table. The group hash table may be extended to 512 entries if RCTRL[GHTX] = 1. Otherwise, an individual address is hashed into the 256-entry individual hash table when RCTRL[GHTX] = 0.

### 15.6.3.7.2 Hash Table Algorithm

The hash table process used in the group hash filtering operates as follows. By default, the Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in IGADDR0–IGADDR7 for individual addresses, and the 256 bits in GADDR0–GADDR7 for group addresses. But in the case where RCTRL[GHTX] is set, both sets of registers are combined into an extended group-only hash table of 512 bits, where IGADDR0–IGADDR7 contain the first 256 bits and GADDR0–GADDR7 contain the last 256 bits. No individual-address table exists in extended mode.

The 48-bit destination address received by the MAC is passed through the Ethernet CRC-32 algorithm to produce a hash value. The CRC polynomial used is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The MAC initializes its CRC register to 0xFFFFFFFF before computing a CRC on the 6 bit-reversed octets of the DA. A non-optimized sample of C code for computing the DA hash is listed in [Figure 15-129](#). The 9 most significant bits of the raw, uninverted CRC are used as the hash table index, H[8:0]. If RCTRL[GHTX] = 0, bits H[8:6] select one of the 8 IGADDR or GADDR registers, while bits H[5:1] select a bit within the 32-bit register. If RCTRL[GHTX] = 1, bits H[8:5] select one of the 16 registers in the {IGADDR, GADDR} set, while bits H[4:0] select a bit within the 32-bit register. For example, if H[8:5] = 7, IGADDR7 is selected, whereas H[8:5] = 9 selects GADDR1.

---

```

/* Wrapper macros for 256-bucket and 512-bucket hash tables:
   Pass 6-byte Ethernet MAC address as parameter. */
#define TSEC_HASH256(macaddr) ((crc32(macaddr) >> 24) & 0xff)
#define TSEC_HASH512(macaddr) ((crc32(macaddr) >> 23) & 0x1ff)

/* CRC constants. Note: CRC-32 polynomial is bit-reversed. */
#define CRC_POLYNOMIAL 0xedb88320
#define CRC_INITIAL    0xffffffff
#define MAC_ADDRLEN    6
#define BITS_PER_BYTE  8

/* crc32() Takes the array of bytes, macaddr[], representing an
   Ethernet MAC address and returns the CRC-32 result over these bytes,
   where each byte is used in bit-reversed form (Ethernet bit order).
   Index 0 of macaddr[] is the first byte of the address on the wire.
   Test case: the result of crc32 on {0x00, 0x01, 0x02, 0x03, 0x04, 0x05}
   should be 0xad0c28f3.
   */
unsigned long crc32(unsigned char macaddr[MAC_ADDRLEN])
{
    unsigned long crc, result;
    int byte, i;

    /* CRC-32 algorithm starts by inverting first 4 bytes */
    crc = CRC_INITIAL;
    /* add each byte to running CRC accumulator */
    for (byte = 0; byte < MAC_ADDRLEN; ++byte) {
        crc ^= macaddr[byte];
        /* shift CRC right to perform but reversal on byte of address */
        for (i = 0; i < BITS_PER_BYTE; ++i)
            if (crc & 1)
                crc = (crc >> 1) ^ CRC_POLYNOMIAL;
            else
                crc >>= 1;
    }
    /* finally, reverse bits of result to get CRC in normal bit order */
    for (result = 0, i = 4*BITS_PER_BYTE-1; i >= 0; crc >>= 1, --i)
        result |= (crc & 1) << i;
    return result;
}

```

---

**Figure 15-129. Sample C Code for Computing eTSEC Hash Table Indices**

If the CRC hash table index selects a bit that is set in the hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the extended hash table prevents roughly 480/512 (93.8%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses. Alternatively, small multicast groups can be held in the exact match MAC address registers, which guarantees that only correct frames are admitted.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 512-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

**NOTE**

The hash table cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. The receive queue filter may be used to reject frames with unintended address hits in the hash table.

**15.6.3.8 Magic Packet Mode**

eTSEC implements the AMD Magic Packet™ specification for LAN-initiated power management. This mode is normally entered with the rest of the system in a low-power sleep mode. Software must enable normal receive function in the Ethernet MAC, and then finally set the MACCFG2[MPEN] bit to enable Magic Packet detection before the system enters a reduced mode. While the rest of the system is operating in low-power mode, the enabled eTSEC continues to receive Ethernet frames, but discards them immediately. Upon receipt of any frame whose contents contain the valid Magic Packet sequence, the eTSEC exits out of Magic Packet mode, thus clearing MACCFG2[MPEN], and raises an error/diagnostic interrupt via IEVENT[MAG], which causes the surrounding system to wake-up. Frames received after Magic Packet mode has exited are received into software buffers as usual. Software can abort Magic Packet mode by writing 0 to MACCFG2[MPEN] at any time.

AMD specify a Magic Packet™ to be any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of frame. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFFFFFFF\_FFFFFFFF, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses. For example, if the station address were 0x112233\_445566, then the MAC would have to receive 0xFFFFFFFF\_FFFFFFFF, 0x112233\_445566, ..., 0x112233\_445566 in any payload to detect a Magic Packet. Only frames addressed specifically to the MAC's station address or a valid multicast or broadcast address can be examined for the Magic Packet sequence.

**15.6.3.9 Flow Control**

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. If the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value.

Table 15-133 lists the flow-control frame structure.

**Table 15-133. Flow Control Frame Structure**

Size [Octets]	Description	Value	Comment
7	Preamble		—
1	SFD		Start frame delimiter
6	Destination address	01-80-C2-00-00-01	Multicast address reserved for use in MAC frames (or MAC station address)
6	Source address		—
2	Length/type	88-08	Control frame type

Table 15-133. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed independent constant of 512 bit-times (unlike slot time). The most-significant octet is transmitted first.
2	Extended MAC parameter		Pause time extended as defined by the PTV[PTE] field. The most significant octet is transmitted first.
40	Reserved	—	—
4	FCS		Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx\_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. Since the pause timer commences counting immediately upon receipt of a PAUSE frame, regardless of whether transmission is currently in progress, a sufficiently large pause time must be received to stop transmission past a frame of MTU size. During a pause, only a control frame can be sent (TCTRL[TFC\_PAUSE] is set). Normal transmission resumes after the pause timer stops counting, or resumes immediately if a pause frame with a zero time-out is received. If another pause-control frame is received during the pause, the period changes to the new value received.

### 15.6.3.10 Interrupt Handling

The following describes what usually occurs within a eTSEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time. There are three kinds of interrupt:
  - Errors (all bits in IEVENT other than RXB, RXF, TXB, or TXF)
  - Receive interrupts, when bits RXB or RXF in IEVENT are set
  - Transmit interrupts, when bits TXB or TXF in IEVENT are set
- Process the TxBDs to reuse them if the IEVENT[TXB, TXF or TXE] were set. Consult register bits TSTAT[TXF0–TXF7] to determine which TxBD rings gave rise to the transmit interrupt in the case of TXF. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the eTSEC; thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set.
- Obtain data from RxBD rings if IEVENT[RXC, RXB or RXF] is set. Consult register bits RSTAT[RXF0–RXF7] to determine which RxBD rings gave rise to the receive interrupt in the case of RXF. If the receive speed is fast or the interrupt delay is long, the eTSEC may have received more than one RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the eTSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch, otherwise a BSY error occurs.

- Clear any set halt or frame interrupt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS] by writing 1s to these bits.
- Continue normal execution.

**Table 15-134. Non-Error Transmit Interrupts**

Interrupt	Description	Action Taken by the eTSEC
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, was updated in one of the enabled TxBD rings.	Programmable 'write with response' TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame from an enabled TxBD ring was transmitted and the last transmit buffer descriptor (TxBD) of that frame was updated.	Programmable 'write with response' to memory on the last TxBD before setting IEVENT[TXF].

**Table 15-135. Non-Error Receive Interrupts**

Interrupt	Description	Action Taken by the eTSEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated in one of the enabled RxBD rings.	Programmable 'write with response' RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received to an enabled RxBD ring and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable 'write with response' to memory on the last RxBD before setting IEVENT[RXF].

### 15.6.3.10.1 Interrupt Coalescing

Interrupt coalescing offers the user the ability to contour the behavior of the eTSEC with regard to frame interrupts. Separate but identical mechanisms exist for both transmitted frames and received frames. In either case, frame interrupts require that software set the I-bit in RxBDs or TxBDs, and disable buffer interrupts (IEVENT[RXB] or IEVENT[TXB]). Particular rings can remain free of interrupts by ensuring that the I-bit is consistently cleared in all BDs. While interrupt coalescing is enabled, a transmit or receive frame interrupt is raised either when a counter threshold-defined number of frames is received/transmitted or the timer threshold-defined period of time has elapsed, whichever occurs first. Disabling and then re-enabling interrupt coalescing forces reset of the coalescing timers and counters to reflect changes made to the threshold registers.

### 15.6.3.10.2 Interrupt Coalescing By Frame Count Threshold

To avoid interrupt bandwidth congestion due to frequent, consecutive interrupts, the user may enable and configure interrupt coalescing to deliberately group frame interrupts, reducing the total number of

interrupts raised. The number of frames received or transmitted prior to an interrupt being raised is determined by the frame threshold field (ICFT) in the appropriate interrupt coalescing configuration register (RXIC or TXIC). The frame threshold field may be assigned a value between 1 and 255. The internal transmit or receive frame counter decrements from this initial value each time a frame is transmitted or received. Upon reaching zero, an interrupt is raised, the appropriate threshold counter is reset to the value in the ICFT field, and then eTSEC continues counting frames while the interrupt is active. The appropriate threshold counter is also reset to the value in the ICFT field if an interrupt is raised subject to the corresponding threshold timer.

### 15.6.3.10.3 Interrupt Coalescing By Timer Threshold

To avoid stale frame interrupts, the user may also assign a timer threshold, beyond which any frame interrupts not yet raised are forced. The timer threshold fields of the receive and transmit interrupt coalescing configuration registers (RXIC[ICTT] and TXIC[ICTT]) are defined in units equivalent to 64 interface clocks or system clocks, depending on the setting of the ICCS field in RXIC and TXIC.

After transmitting a frame, the transmit interrupt coalescing threshold time begins counting down from the value in TXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful transmit stop completion before the coalescing timer expires, the eTSEC will issue two interrupts, the first for GTS, the second for TXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GTS event, it is recommended that the user mask out the TXF event during execution of the service routine. After receiving a frame, the receive interrupt coalescing threshold time begins counting down from the value in RXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful receive stop completion before the coalescing timer expires, the eTSEC will issue two interrupts, the first for GRS, the second for RXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GRS event, it is recommended that the user mask out the RXF event during execution of the service routine.

The interrupt coalescing timer thresholds (transmit and receive, operating independently) may be values ranging from 0x0001 to 0xFFFF. [Table 15-136](#) specifies the range of possible timing thresholds subject to timer clock source, the interface or system frequency, and the value of the RXIC[ICTT] or TXIC[ICTT] field.

**Table 15-136. Interrupt Coalescing Timing Threshold Ranges**

ICCS (Clock Source)	eTSEC Interface Format and Frequency or eTSEC System Frequency	Interrupt Coalescing Threshold Time	
		Minimum (ICTT = 0x0001)	Maximum (ICTT = 0xFFFF)
0 (I/F clock)	10Base-T at 2.5 MHz	25.6 $\mu$ s	1.68 s
0 (I/F clock)	100Base-T at 25 MHz	2.56 $\mu$ s	168 ms
0 (I/F clock)	1000Base-T at 125 MHz	0.51 $\mu$ s	33.6 ms
1 (sys. clock)	eTSEC operating at 266 MHz	0.24 $\mu$ s	15.7 ms
1 (sys. clock)	eTSEC operating at 333 MHz	0.19 $\mu$ s	12.6 ms

The transmit timer threshold counter is reset to the value in TXIC[ICTT] and begins counting down on transmission of the frame following an interrupt.



The receive timer threshold counter is reset to the value in RXIC[ICTT] and begins counting down on receiving the frame following an interrupt.

### 15.6.3.11 Inter-Frame Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-frame gap). After a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the frame is not successfully sent within a specified number of retries, an error is indicated.

The minimum inter-frame gap time for back-to-back transmission is 96 bit times. The receiver receives back-to-back frames with this minimum spacing. In addition, after waiting a required number of bit times (based on the back-off algorithm), the transmitter waits for carrier sense to be negated before retransmitting the frame. Retransmission begins 36 bit times after carrier sense is negated for at least 60 bit times.

### 15.6.3.12 Internal and External Loop Back

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default. Clearing this bit results in normal operation.

### 15.6.3.13 Error-Handling Procedure

The eTSEC reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in [Table 15-137](#).

**Table 15-137. Transmission Errors**

Error	Response
Transmitter underrun	Transmitter underrun can occur either after frame transmission has commenced, or in response to an incomplete sequence of TxBDs. In the former case, the controller sends 32 bits that ensure a CRC error, and terminates buffer transmission. In the latter case, the relevant transmit queue is halted. In all cases, the eTSEC closes the buffer, sets TxBD[UN], IEVENT[XFUN], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, IEVENT[CRL], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, IEVENT[LC], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).

**Table 15-137. Transmission Errors (continued)**

Error	Response
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error, and therefore TxBD[UN] is set, but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Data parity error	Data in the transmit FIFO was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues transmission until halted explicitly.
Babbling transmit error	A frame is transmitted which exceeds the MAC's Maximum Frame Length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption.

Reception errors are described in [Table 15-138](#).

**Table 15-138. Reception Errors**

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxB[OV], sets RxB[L], closes the buffer, increments the discarded frame counter (RDRP), and sets IEVENT[RXF]. The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY] and increments the discarded frame counter (RDRP). In addition, the RSTAT[QHLT $n$ ] bit is set. RDRP will increment for each frame that is received while the receiver is halted due to a busy condition. The halted queue resumes reception once the RSTAT[QHLT $n$ ] bit is cleared.
Filed frame to invalid queue error	A frame is received and discarded as a result of the filer directing it to an RxB ring that is currently not enabled. The controller sets IEVENT[FIQ] and increments the discarded frame counter (RDRP).
Parser error	If the receive frame parser is enabled, a parse error can be flagged as a result of inconsistencies discovered between fields of the embedded packet headers. For example, the L2 header may indicate an IPv4 header, but the IP version number fails to match. In the event of a parse error, parsing is terminated at the inconsistent header, and the RxFCB[PERR] field indicates at which layer of the protocol stack the error was discovered. Receiver function continues regardless of parse errors, but IEVENT[PERR] is set. The receive queue filer may operate with reduced or default information in some cases; therefore, filer rule sets should be constructed so as to be tolerant of malformed frames. <b>Note:</b> Any values in the length/type field between 1500 and 1536 will be treated as a length, however, only illegal packets exist with this length/type since these are not valid lengths and not valid types. These are treated by the MAC logic as out of range. Software must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxB[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The eTSEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.
CRC error	If a CRC error occurs, the controller sets RxB[CR], closes the buffer, and sets IEVENT[RXF]. This eTSEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.

**Table 15-138. Reception Errors (continued)**

Error	Description
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame and increments the discarded frame counter (RDRP). In addition the RSTAT[QHLT $n$ ] bit is set. The halted queue resumes reception once the RSTAT[QHLT $n$ ] bit is cleared.
Data parity error	Data in the receive FIFO or filter table was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues reception until halted explicitly.
Babbling receive error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues.

## 15.6.4 TCP/IP Off-Load

Each eTSEC provides hardware support for accelerating the basic functions of TCP/IP packet transmission and reception. By default, these features are disabled and must be explicitly enabled via RCTRL and TCTRL. In this configuration, the eTSEC processes frames as vanilla Ethernet frames and none of the multi-ring QoS/CoS receive services or per-frame VLAN insertion and deletion are available. Operate eTSEC in this default configuration when using existing TCP/IP stack software that has not been modified to take advantage of TOE.

TOE can be enabled independently for Rx and Tx and at various levels. Receive TOE functions are controlled by RCTRL and transmit functions via a combination of TCTRL[TUCSEN] and the Tx frame control block.

On receive, according to RCTRL[PRSDEP], eTSEC can parse frames at layer 2 of the stack only (Ethernet headers and switching headers), layers 2 to 3 (including IPv4 or IPv6), or layers 2 to 4 (including TCP and UDP). TOE provides protocol header recognition, header verification (IPv4 header checksum verification), and TCP/UDP payload checksum verification including verification of associated pseudo-header checksums. For large frames off-load of checksum verification saves a significant fraction of the CPU cycles that would otherwise be spent by the TCP/IP stack. IP packet fragmentation and re-assembly, and TCP stream establishment and tear-down are not performed in hardware. The frame parser sets RQFPR[IPF] status flag encountering a fragmented frame. The frame parser in eTSEC searches a maximum of 512 bytes from the start of a received frame when attempting to locate headers; headers deeper than 512 bytes are assumed not to exist, and any associated receive status flags in the frame control block remain cleared.

On transmit, TOE provides IPv4 and TCP/UDP header checksum generation. Like receive TOE, checksum generation reduces CPU load significantly for TCP/IP stacks modified to exploit eTSEC TOE functions. The eTSEC does not checksum transmitted packets with IPv6 routing headers or calculate TCP/UDP checksums from IP fragments. If a transmitted TCP segment requires checksum generation but IPv6 extension headers would prevent eTSEC from calculating the pseudo-header checksum, software can calculate just the pseudo-header checksum in advance and supply it to the eTSEC as part of per-frame TOE configuration.

### 15.6.4.1 Frame Control Blocks

Frame control blocks (FCBs) are 8-byte blocks of TOE control and/or status data that are passed between software (driver and TCP/IP stack) and each eTSEC. A FCB always precedes the frame it applies to, and is present only when TOE functions are being used. As Figure 15-130 shows, the first BD of each frame points to the initial data buffer and the FCB. The initial data buffer must be at least 8 bytes long to contain the FCB without breaking it. Custom or received Ethernet preamble sequences also follow the FCB if preambles are visible.

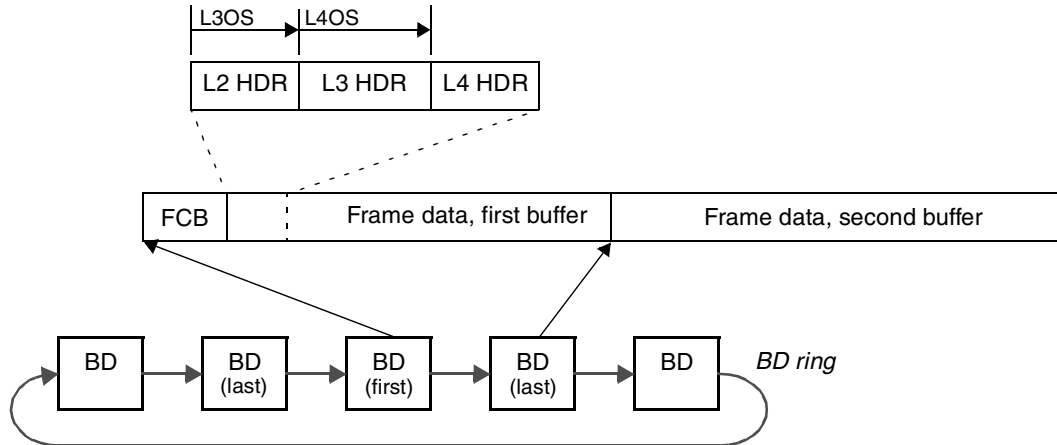


Figure 15-130. Location of Frame Control Blocks for TOE Parameters

For TxBD rings, FCBs are assumed present when the TxBD[TOE/UN] bit is set by user software. The eTSEC ignores the TxBD[TOE/UN] bit in all BDs other than those pointing to initial data buffers, therefore FCBs must not be inserted in second and subsequent data buffers. Since TxBD[TOE/UN] can be set under software discretion, TOE acceleration for transmit may be applied on a frame-by-frame basis.

In the case of RxBD rings, FCBs are inserted by the eTSEC whenever RCTRL[PRSDEP] is set to a non-zero value. Only one FCB is inserted per frame, in the buffer pointed to by the RxBD with bit F set. TOE acceleration for receive is enabled for all frames in this case.

### 15.6.4.2 Transmit Path Off-Load

TOE functions for transmit are defined by the contents of the Tx FCB. Figure 15-131 describes the definition for the Tx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	UDP	CIP	CTU	NPH								
Offset + 2	L4OS								L3OS							
Offset + 4	PHCS															
Offset + 6	VLCTL															

Figure 15-131. Transmit Frame Control Block

The contents of the Tx FCB are defined in [Table 15-139](#).

**Table 15-139. Tx Frame Control Block Descriptions**

Bytes	Bits	Name	Description
0–1	0	VLN	VLAN control word valid. 0 Ignore VLCTL field. 1 If VLAN tag insertion is enabled for eTSEC, use the VLCTL field as the VLAN control word.
	1	IP	Layer 3 header is an IP header. 0 Ignore layer 3 and higher headers. 1 Assume that the layer 3 header is an IPv4 or IPv6 header, and take L3OS field as valid.
	2	IP6	IP header is IP version 6. Valid only if IP = 1. 0 IP header version is 4. 1 IP header version is 6.
	3	TUP	Layer 4 header is a TCP or UDP header. 0 Do not process any layer 4 header. 1 Assume that the layer 4 header is either TCP or UDP (see UDP bit), and offload checksumming on the basis that the IP header has no extension headers.
	4	UDP	UDP protocol at layer 4. 0 Layer 4 protocol is either TCP (if TUP = 1) or undefined. 1 Layer 4 protocol is UDP if TUP = 1.
	5	CIP	Checksum IP header enable. 0 Do not generate an IP header checksum. 1 Generate an IPv4 header checksum.
	6	CTU	Checksum TCP or UDP header enable. 0 Do not generate a TCP or UDP header checksum. RFC 768 advises that UDP packets not requiring checksum validation should have their checksum field set to zero. 1 Generate a TCP header checksum if IP = 1 and TUP = 1 and UDP = 0.
0–1	7	NPH	Disable calculation of TCP or UDP pseudo-header checksum. This bit should be set if IP options need to be consulted in forming the pseudo-header checksum, as eTSEC does not examine IP options or extension headers for TCP/IP offload on transmit. 0 Calculate TCP or UDP pseudo-header checksum as normal, assuming that the IP header has no options. 1 Do not calculate a TCP or UDP pseudo-header checksum, but instead use the value in field PHCS when determining the overall TCP or UDP checksum.
	8–15	—	Reserved
2–3	0–7	L4OS	Layer 4 header offset from start of layer 3 header. The layer 4 header starts L4OS octets after the layer 3 header if it is present. The maximum layer 3 header length supported is thus 255 bytes, which may prevent TCP/IP offload on particularly large IPv6 headers.
	8–15	L3OS	Layer 3 header offset from start of frame not including the 8 bytes for this FCB. The layer 3 header starts L3OS octets from the start of the frame including any custom preamble header that may be present. The maximum layer 2 header length supported is thus 255 bytes.
4–5	0–15	PHCS	Pseudo-header checksum (16-bit one's complement sum with carry wraparound, but without result inversion) for TCP or UDP packets, calculated by software. Valid only if NPH = 1.
6–7	0–15	VLCTL	VLAN control word for insertion in the transmitted VLAN tag. Valid only if VLN = 1.

### 15.6.4.3 Receive Path Off-Load

Upon receive, the Rx FCB returns the status of frame parse and TOE functions applied to the accompanying frame. [Figure 15-132](#) describes the definition for the Rx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	CIP	CTU	EIP	ETU					PERR			
Offset + 2				RQ				PRO								
Offset + 4																
Offset + 6	VLCTL															

**Figure 15-132. Receive Frame Control Block**

The contents of the Rx FCB are defined in [Table 15-140](#).

**Table 15-140. Rx Frame Control Block Descriptions**

Bytes	Bits	Name	Description
0–1	0	VLN	VLAN tag recognized. This bit is set only if RCTRL[VLEX] is set. 0 No VLAN tag recognized. 1 IEEE Std. 802.1Q VLAN tag found; VLAN control word in VLCTL is valid.
	1	IP	IP header found at layer 3. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IP discovery. See also IP6 bit of FCB. 0 No layer 3 header recognized. 1 An IP header was recognized at layer 3; the IANA protocol identifier for the next header can be found in PRO; see PRO for more information.  If S/W is relying on the RxFCB for the parse results, any RxFCB[IP] bits set with the corresponding RxFCB[PRO] = 0xFF indicates a fragmented packet (or that this packet had a back-to-back IPv6 routing extension header). Additionally, RQFPR[IPF] (see <a href="#">Section 15.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR)”</a> ) indicates that the packet was fragmented.
	2	IP6	IP version 6 header found at layer 3. 0 No IPv6 header was found. 1 The layer 3 header was an IPv6 header provided IP = 1.
	3	TUP	TCP or UDP header found at layer 4. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable TCP/UDP discovery. 0 No layer 4 header recognized. 1 The layer 4 header was recognized as either TCP (PRO = 0x06) or UDP (PRO = 0x11).
	4	CIP	IPv4 header checksum checked. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IPv4 checksum verification. 0 IPv4 header checksum not verified, either because verification was disabled or a valid IPv4 header could not be located. 1 IPv4 header checksum was verified by the eTSEC, and bit EIP indicates result.
	5	CTU	TCP or UDP header checksum checked. RCTRL[PRSDEP] must be set to 11 in order to enable layer 4 checksum verification. 0 TCP or UDP header checksum not verified, either because verification was disabled or a valid TCP or UDP header could not be located. If a UDP header with zero checksum was located, this bit is cleared in accordance with RFC 768. 1 TCP or UDP header checksum was verified by the eTSEC, and ETU indicates result.
	6	EIP	IPv4 header checksum verification error. Not valid unless CIP = 1. 0 No checksum error in IPv4 header. 1 Error in header checksum only if IP = 1 and IP6 = 0.
0–1	7	ETU	TCP or UDP header checksum verification error. Not valid unless CTU = 1. 0 No checksum error in TCP or UDP header. 1 Error in header checksum only if PRO = 0x06 or PRO = 0x11.
	8–11	—	Reserved
	12–13	PER	Parse error. 00 No error in L2 to L4 parse 01 Reserved 10 Inconsistent or unsupported L3 header sequence 11 Reserved
	14–15	—	Reserved

Table 15-140. Rx Frame Control Block Descriptions (continued)

Bytes	Bits	Name	Description
2–3	0–1	—	Reserved
	2–7	RQ	Receive queue index. This index was selected by the eTSEC Rx Filer (from a matching Filer rule's RQCTRL[Q] field) when it accepted the associated frame. If filing is not enabled, RQ is zero. Note that the 3 least significant bits of RQ correspond with the RxBD ring index whenever RCTRL[FSQEN] = 0.
	8–15	PRO	<p>If IP = 1, PRO is set as follows:</p> <ul style="list-style-type: none"> <li>• PRO=0xFF for a fragment header or a back to back route header</li> <li>• PRO=0xnn for an unrecognized header, where nn is the next protocol field</li> <li>• PRO=(TCP/UDP header), as defined in the IANA specification, if TCP or UDP header is found</li> </ul> <p>If IP = 0, PRO is undefined.</p> <p>Note that the eTSEC parser logic stops further parsing when encountering an IP datagram that has indicated that it has fragmented the upper layer protocol. This in general means that there is likely no layer 4 header following the IP header and extension headers. eTSEC leaves the RxFCB[PRO] and RQFPR[L4P] fields 0xFF in this case, which usually means that there was no IP header seen. In this case RxFCB[IP] and optionally RxFCB[IP6] will be set. IP header checksumming will operate and perform as intended. Most of the time, the eTSEC will update the RxFCB[PRO] field and RQFPR[L4P] fields with whatever value was found in the protocol field of the IP header. See <a href="#">Section 15.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR),”</a> for a description of RQFPR.</p>
4–5	0–15	—	Reserved
6–7	0–15	VLCTL	VLAN control word as per IEEE Std. 802.1Q. The lower 12 bits comprise the VLAN identifier. Valid only if VLN = 1.

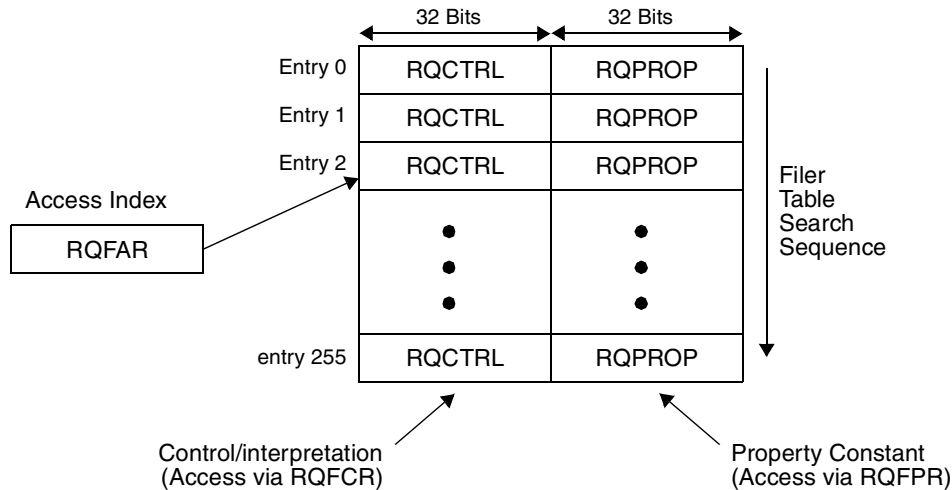
## 15.6.5 Quality of Service (QoS) Provision

This section describes the quality of service support features of this device.

### 15.6.5.1 Receive Queue Filer

The receive queue filer receives protocol header properties extracted from the incoming frame by the eTSEC frame parse engine. A property is defined to be a field extracted from a packet header, such as a TCP port number or VLAN identifier. As soon as the last identifiable header has been recognized, the filer commences searching the receive queue filer table, comparing properties in the table against properties extracted from the frame. This table is illustrated in [Figure 15-133](#). Software populates the table with property values, stored to the RQPROP field, and indicates how to match and interpret the properties by setting flags in the RQCTRL field. The eTSEC memory map provides access to these fields by way of an address register (RQFAR) and two porthole registers (RQFCR and RQFPR).





**Figure 15-133. Structure of the Receive Queue Filer Table**

### 15.6.5.1.1 Filing Rules

Unless the filer is disabled, every received frame from the Ethernet MAC or FIFO interface initiates a search of the receive queue filer table, starting at entry 0. The table search is terminated as soon as an entry is found whose contents match a property of the frame. Accordingly, software must guarantee that at least one entry will result in a match—even if only to set a default receive queue index.

Since eTSEC searches the table at a rate of two entries every system clock cycle, all 256 entries can be searched in the time taken to receive a 64-byte Ethernet frame.

Each entry of the receive queue filer table specifies a simple match rule for determining how to process the received frame. The elements of a filing rule, expressed in the RQCTRL and RQPROP fields, are summarized as follows:

- The PID field in RQCTRL identifies what property is being matched against RQPROP. The eTSEC supports 16 properties, some of which are different portions of the same header field. Reserved or unused bits in RQPROP are read as zero. See [Section 15.5.3.3.8, “Receive Queue Filer Table Property Register \(RQFPR\),” on page 15-58](#) for a list of all properties and their associated PID values.
- The Q field in RQCTRL identifies which one of 64 virtual receive queues the frame should be filed to (sent via DMA) in the event of a filing rule match that accepts the frame. The physical RxB ring this queue maps to is controlled by the RCTRL[FSQEN] bit. If RCTRL[FSQEN] = 0, the three least significant bits of the Q field indicate which physical RxB ring hosts the queue. If RCTRL[FSQEN] = 1, RxB ring 0 hosts all receive queues, but the RxFCB[RQ] field allows software to distinguish queues by ID. In all cases if Q maps to a RxB ring that is not currently enabled, the frame is discarded with an IEVENT[FIQ] error.
- The REJ field in RQCTRL controls whether the frame is to be rejected (REJ = 1) or filed (REJ = 0) upon a filing rule match. Rejected frames occupy Rx FIFO space, but do not consume memory bus cycles.
- The CMP field in RQCTRL determines how property PID is compared against RQPROP. Equality, inequality, greater-or-equal, and less-than compares are available.

- The AND field in RQCTRL allows more than one comparison in a sequence to be chained together as a Boolean AND condition. Setting AND = 1 defers evaluation of the rule until the next entry has been matched, which may, in turn, have AND set. If any comparison involving AND = 1 fails, the entire chained sequence fails. A typical use for AND is to combine a pair of comparisons in a range match; the first such entry has AND = 1, the second has AND = 0 and its values of Q and REJ take effect.
- The CLE field in RQCTRL offers a way to bracket a set of consecutive—perhaps related—rules into a rule cluster. A cluster must be preceded by a guard rule, which simply determines whether the cluster rules can be evaluated. If the guard rule succeeds and its last entry has both CLE = 1 and AND = 1, the cluster rules that follow are enabled. The cluster ends at the first entry where CLE = 1 and AND = 0, which may also belong to a rule that files or rejects a frame. If the guard rule fails, all rules in the cluster are skipped, including mask\_register assignments. Clusters must not be nested.

### 15.6.5.1.2 Comparing Properties with Bit Masks

By default, extracted properties are compared arithmetically according to the CMP field in each RQCTRL word. This permits point value matches in each table entry, and range checks across a pair of table entries combined with the AND attribute in RQCTRL. However, inspection of the parse flags, Ethernet preamble, and IP addresses typically requires ‘don’t care’ bit fields in the properties to be cleared as part of the comparison. The eTSEC provides a dedicated 32-bit register, known as the mask\_register, for performing such masking operations. At the start of each table search by the filer, mask\_register is reset to 0xFFFF\_FFFF, which ensures that no masking occurs.

Filer rules may be configured to assign specific bit patterns to mask\_register. Such rules can be configured to either match always (useful for implementing a default rule and specifying an associated receive queue), or fail always (which prevents termination of the filer table search). Once mask\_register has been assigned, it retains its value until it is reassigned or the table search terminates. All properties are non-destructively bit-wise ANDed with mask\_register prior to comparison in subsequent rules, which allows an entire cluster of rules to make use of a common mask. Individual masks for specific rules can also be created simply by combining a mask\_register assignment (match always form) with a regular rule using the AND attribute.

To create a mask\_register assignment rule, it is necessary to select PID = 0 in RQCTRL, and choose CMP such that the rule either matches (CMP = 01) or fails (CMP = 11). In this entry, RQPROP is then considered to be the assigned bit vector.

### 15.6.5.1.3 Special-Case Rules

It is frequently useful to create rules that are guaranteed to succeed or fail, specifically to enforce a default filing decision or act as null entries. Suggested constructions for such rules are shown in [Table 15-141](#).

**Table 15-141. Special Filer Rules**

Rule Description	RQCTRL Fields						RQPROP Word	RQCTRL Word <sup>1</sup>
	CLE	REJ	AND	Q	CMP	PID		
Default file—Always file frame to ring Q	0	0	0	Q	01	0000	0x0000_0000	0x0000_0q20
Default reject—Always discard frame	0	1	0	000_000	01	0000	0x0000_0000	0x0000_0120
Empty rule in AND—Always matches	0/1 <sup>2</sup>	0	1	000_000	01	0000	0xFFFF_FFFF	0x0000_00A0
Empty rule in rule set—Always fails	0/1 <sup>3</sup>	0	0	000_000	11	0000	0xFFFF_FFFF	0x0000_0060

<sup>1</sup> Hexadecimal digits *qq* denotes field Q shifted left 2 bits.

<sup>2</sup> Set CLE = 1 if the empty rule guards a cluster.

<sup>3</sup> Set CLE = 1 if the empty rule occurs at the end of a cluster.

### 15.6.5.1.4 Filer Interrupt Events

The filer can produce two interrupt events in IEVENT. Event FIR indicates an error condition where the filer was unable to provide a definite result, either because no rule in the table succeeded, or because frames arrived too rapidly to complete searching of the table. Event FIQ indicates that the filer accepted a frame to a RxBD ring that was not enabled in RQCTRL (this can also occur if the filer is disabled, but RxBD ring 0—default queue or FSQEN mode queue—is not enabled). FIQ is also asserted in the case where no rule in the entire table succeeded. The various combinations of these interrupt events and their interpretation appear in [Table 15-142](#).

**Table 15-142. Receive Queue Filer Interrupt Events**

IEVENT[FIR]	IEVENT[FIQ]	Description
0	0	No error. The filer successfully rejected or filed a frame.
0	1	Illegal queue error. The filer accepted a frame to a RxBD ring that is disabled (including ring 0 if filing is disabled).
1	0	Partial search error. The filer did not have sufficient time to complete its search of the filer table.
1	1	No matching rule error. The filer searched all 256 entries of the filer table without finding a rule that succeeds.

### 15.6.5.1.5 Setting Up the Receive Queue Filer Table

The eTSEC frame parser always provides values for all properties, even where the relevant headers are not available. In the latter case, the filer is given default properties that can be used to avoid conflict with normal, defined property values. Accordingly, the rules in the filer table can be partitioned into rule sets such that if all rules in a given set fail (due to headers being unavailable), lower priority rule sets can be subsequently searched until either a rule set provides a match or a single default—catch-all—rule specifies a definite receive queue. For example, an 802.1p priority rule set may be followed by an IP TOS rule set,

followed by a default rule; thus, if no VLAN tag appears in the received frame, the TOS rules are checked, or the default is activated should no IP header be present.

The rule cluster feature is used to conditionalize evaluation of rule sets. Typically, this avoids evaluating rules based on properties that may not be valid or relevant to the filing or filtering decision. For example, TCP-related rules might be clustered behind a guard rule that checks that a TCP header has appeared and the IP address matches our home address. Property 1—the parse flags property—is provided specifically to check the characteristics of the received frame and the parser error status. The mask\_register is typically assigned beforehand to extract specific flags, in which case care should be taken that mask\_register be reassigned an appropriate mask vector for following comparisons.

In many cases it will be possible to write the entire filer table before using eTSEC, as the rule set is static. However, dynamic rule updates can be supported by pre-allocating partially instantiated rule sets, which software rewrites as necessary. Rules that are not instantiated should be composed of empty entries, as indicated in [Table 15-141](#). In many cases empty entries can be overwritten by software without stopping eTSEC's receive function.

#### 15.6.5.1.6 Filer Example—802.1p Priority Filing

This example, shown in [Table 15-143](#), illustrates how to file frames according to layer 2 802.1p priority. This matches against property 1001, comparing each specific priority level in order to associate them with a RxBD ring index. Note that if a VLAN tag does not appear in the frame, the parser passes priority 0 to the filer, which will always match the rule at entry 7 and terminate the table search.

**Table 15-143. Filer Table Example—802.1p Priority Filing**

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	000_000	00	1001	0x0000_0007	File priority 7 to ring 0	0x0000_0009
1	0	0	0	000_001	00	1001	0x0000_0006	File priority 6 to ring 1	0x0000_0409
2	0	0	0	000_010	00	1001	0x0000_0005	File priority 5 to ring 2	0x0000_0809
3	0	0	0	000_011	00	1001	0x0000_0004	File priority 4 to ring 3	0x0000_0C09
4	0	0	0	000_100	00	1001	0x0000_0003	File priority 3 to ring 4	0x0000_1009
5	0	0	0	000_101	00	1001	0x0000_0002	File priority 2 to ring 5	0x0000_1409
6	0	0	0	000_110	00	1001	0x0000_0001	File priority 1 to ring 6	0x0000_1809
7	0	0	0	000_111	00	1001	0x0000_0000	File undefined 802.1p or priority 0 to ring 7—Default always matches	0x0000_1C09

#### 15.6.5.1.7 Filer Example—IP Diff-Serv Code Points Filing

This example demonstrates use of rule priority for determining class selector codepoints (RFC 2474) from the IP TOS property. An example filer table is shown in [Table 15-144](#). The example relies on the fact that the first rule matched terminates the search, hence successively lower Diff-Serv codepoint ranges can be compared in each step until the default (zero or greater) range is reached. By default, property 1010 (IP

TOS) takes the value 0x00 if no IP headers were recognized, therefore the table search will always terminate.

**Table 15-144. Filer Table Example—IP Diff-Serv Code Points Filing**

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	001_000	01	1010	0x0000_00E0	File class 7 to queue 8 (TOS >= 0xE0)	0x0000_202A
1	0	0	0	001_001	01	1010	0x0000_00C0	File class 6 to queue 9 (TOS >= 0xC0)	0x0000_242A
2	0	0	0	001_010	01	1010	0x0000_00A0	File class 5 to queue 10 (TOS >= 0xA0)	0x0000_282A
3	0	0	0	001_011	01	1010	0x0000_0080	File class 4 to queue 11 (TOS >= 0x80)	0x0000_2C2A
4	0	0	0	000_100	01	1010	0x0000_0060	File class 3 to queue 4 (TOS >= 0x60)	0x0000_102A
5	0	0	0	001_100	01	1010	0x0000_0040	File class 2 to queue 12 (TOS >= 0x40)	0x0000_302A
6	0	0	0	010_100	01	1010	0x0000_0020	File class 1 to queue 20 (TOS >= 0x20)	0x0000_502A
7	0	0	0	011_100	01	1010	0x0000_0000	File class 0 to queue 28 (TOS >= 0x00) or file to ring 4 by default	0x0000_702A

### 15.6.5.1.8 Filer Example—TCP and UDP Port Filing

This example demonstrates rule clusters and AND-combined entries for filing packets based on transport protocol and well-known port numbers in a termination application. An example filer table is shown in [Table 15-145](#). The example contains two clusters; the first is entered only for TCP packets, the second is entered only for UDP packets. A default filing rule catches the case where neither TCP nor UDP headers are found. Each cluster compares source port number (property 1111) against a list of server ports, and files the packets accordingly. Note that entries 1 and 2 form an AND rule for checking that the port number >= 20 and port number < 22. Entries 4 and 5 are initially set up to always fail (zero port number), and thus comprise empty entries that can be used at a later time.

**Table 15-145. Filer Table Example—TCP and UDP Port Filing**

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	1	0	1	000_000	00	1011	0x0000_0006	Enter cluster if layer 4 is TCP	0x0000_028B
1	0	0	1	000_000	01	1111	0x0000_0014	AND rule—FTP from TCP ports 20 and 21: file to ring 2	0x0000_00AF
2	0	0	0	000_010	11	1111	0x0000_0016		0x0000_086F
3	0	0	0	000_011	00	1111	0x0000_0017	telnet from TCP port 23: file to ring 3	0x0000_0C0F
4	0	0	0	000_000	00	1111	0x0000_0000	<i>empty entry reserved for future use</i>	0x0000_000F
5	0	0	0	000_000	00	1111	0x0000_0000	<i>empty entry reserved for future use</i>	0x0000_000F
6	1	0	0	000_001	01	0000	0x0000_0000	end cluster; default TCP: file to ring 1	0x0000_0620
7	1	0	1	000_000	00	1011	0x0000_0011	Enter cluster if layer 4 is UDP	0x0000_028B
8	0	0	0	000_101	00	1111	0x0000_0801	NFS from UDP port 2049	0x0000_140F

Table 15-145. Filer Table Example—TCP and UDP Port Filing (continued)

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
9	0	0	0	000_111	00	1111	0x0000_0208	Route from UDP port 520	0x0000_000F
10	0	0	0	000_110	00	1111	0x0000_0045	TFTP from UDP port 69	0x0000_180F
11	1	0	0	000_100	01	0000	0x0000_0000	End cluster; default UDP: file to ring 4	0x0000_1220
12	0	0	0	000_000	01	0000	0x0000_0000	By default, file to ring 0	0x0000_0020

### 15.6.5.2 Transmission Scheduling

Each eTSEC can maintain multiple TxBD rings (or transmission queues) to satisfy QoS requirements. The ability to choose from a number of transmission streams dynamically is especially important during periods of network congestion. Certain application such as voice and video streaming are delay sensitive, but loss insensitive. For instance, VoIP applications require little bandwidth, but are highly sensitive to latency. Conversely, FTP or SMTP protocols are delay insensitive, but loss sensitive.

eTSEC has a transmission scheduler that implements a programmable QoS regime. The scheduler is responsible for choosing which of the prefetched TxBDs shall be processed next, and accordingly issuing DMA requests to service the data stream described by the chosen BD(s). The scheduler cycle is one of:

1. decide on a TxBD queue,
2. transmit exactly one frame from that queue, and
3. return to deciding on another queue, in step 1.

If TCTRL[TXSCHED] is set to 00, no transmission scheduling occurs, and only TxBD ring 0 is polled for new data to transmit, with DMACTRL controlling waiting or polling. TCTRL[TXSCHED], if not zero, can be programmed to invoke one of two scheduling algorithms, namely priority-based queuing (PBQ), and modified weighted round-robin queuing (MWRR). In all cases where TCTRL[TXSCHED] is not zero, the scheduler can choose from among 1 to 8 TxBD rings per eTSEC, with individual rings being enabled by the setting of TQUEUE[EN0–EN7] bits. For example, TxBD rings 3, 4, and 7 may be enabled for scheduling by setting EN3, EN4, and EN7, and clearing all other EN bits.

#### 15.6.5.2.1 Priority-Based Queuing (PBQ)

PBQ is the simplest scheduler decision policy. The enabled TxBD rings are assigned a priority value based on their index. Rings with a lower index have precedence over rings with higher indices. For example, TxBD ring 0 has higher priority than TxBD ring 1, and TxBD ring 1 has higher priority than TxBD ring 2, and so on.

The scheduling decision is then achieved as follows:

```

loop
  priority_ring = null;
  ring = 0;
  while priority_ring == null and ring <= 7 loop
    if enabled(ring) and not ring_empty(ring) then
      priority_ring = ring;

```

```

        endif
        ring = ring + 1;
    endloop
    if priority_ring >= 0 then
        while not ring_empty(priority_ring) loop
            transmit_frame(priority_ring);
        endloop
    endif
endloop

```

In practice a protocol stack or device driver can abuse PBQ by attempting to queue too much traffic onto high priority rings. It is recommended that the highest priority ring should normally not be used at all except for frames requiring the utmost urgent transmission. This allows emergency traffic to overtake backlogged queues out of sequence.

### 15.6.5.2.2 Modified Weighted Round-Robin Queuing (MWRR)

eTSEC implements a modified weighted round-robin scheduling algorithm across all enabled TxBD rings when TCTRL[TXSCHEd] = 10. In MWRR, the weights in the TR03WT and TR47WT registers determine the ideal size of each transmit slot, as measured in multiples of 64 bytes. Thus, to set a transmit slot of 512 bytes, a weight of 512/64 or 8 needs to be set for the ring. In this mode TxBD rings 1–7 are selected in round-robin fashion, whereas TxBD ring 0, if enabled with ready data for transmission, is always selected in between other rings so as to expedite transmission from ring 0.

The scheduling decision is then achieved as follows:

```

for ring = 1..7 and enabled(ring) loop
    credit[ring] = 0;
endloop
for ring = 1..7 and enabled(ring) loop
    if not ring_empty(0) then
        credit[0] = credit[0] + weight[0];
        while credit[0] > 0 loop
            transmit_frame(0);
            credit[0] = credit[0] - frame_size;
            if ring_empty(0) then
                credit[0] = 0;
            endif
        endloop
    endif
    if not ring_empty(ring) then
        credit[ring] = credit[ring] + weight[ring];
    endif
    while credit[ring] > 0 loop
        transmit_frame(ring);
        credit[ring] = credit[ring] - frame_size;
        if ring_empty(ring) then
            credit[ring] = 0;
        endif
    endloop
endloop
endloop

```

The algorithm checks registers TQUEUE[EN0–EN7] for `enabled()`, TSTAT[THLT0–THLT7] for `ring_empty()`, and TRxWT for `weight()`. For TxBD ring  $k$ , having a weight  $WT_k$ , the long term average throughput for that ring is:

$$\text{rate of queue}[k] \text{ (K = 1 to 7)} = (\text{available bandwidth}) * WT_k / (\text{sum}(WT_i) + 6WT_0)$$

$$\text{rate of queue}(0) = (\text{available bandwidth}) * 7 * WT_0 / (\text{sum}(WT_i) + 6WT_0)$$

where  $i = 0$  to 7

## 15.6.6 Lossless Flow Control

The eTSEC DMA subsystem is designed to be able to support simultaneous receive and transmit traffic at gigabit line rates. If the host memory has sufficient bandwidth to support such line rates, then the principle cause of overflow on receive traffic will be due to a lack of Rx BDs. Thus, the long term receive throughput will be determined by the rate at which software can process receive traffic. If a user desires to prevent dropped packets, they can inform the far-end link to stop transmission while the software processing catches up with the backlog. The current hardware initiated back-pressure mechanism (the RX\_PAUSE register) is designed to allow for momentary backlogging of the Rx DMA (due to host memory contention), not a lack of Rx BDs.

To avoid overflow in the latter case, back pressure must be applied to the far-end transmitter before the Rx descriptor controller encounters a non-empty BD and halts with a BSY error. As there is lag between application of back-pressure and response of the far-end, the pause request must be issued while there are still BDs free in the ring. In the traditional eTSEC descriptor ring programming model, there is no way for hardware to know how many free BDs are available, so software must initiate any pause requests required during operation. If software is backlogged, the request may not be issued in time to prevent BSY errors. To allow the eTSEC to generate the pause request automatically, additional information (a pointer to the last free BD and ring length) is required.

### 15.6.6.1 Back Pressure Determination via Free Buffers

Ultimately, the rate of data reception is determined by how quickly software can release buffers back into the receive ring(s). Each time a buffer is freed, the associated BD has its empty bit set and hardware is free to consume both. Thus the number of free BDs in a given Rx ring indicates how close hardware is to the end of that ring. To prevent data loss, back pressure should be applied when the number of free BDs drops below some critical level. The number of BDs that can be consumed by an incoming packet stream while back-pressure takes effect is determined by several factors, such as: receive traffic profile, transmit traffic profile, Rx buffer size, physical transmission time between eTSEC and far-end device and intra-device latency. Theoretically, the worst case will be:

$$\text{FreeBDsRequired} = \frac{\text{MaxFrameSize}}{\text{MinFrameSize} + \text{IFG}} + \frac{\text{MaxFrameSize}}{\text{RxBufferSize}} + \text{LinkDelay}$$



This case comes about when:

- The eTSEC has just started transmitting a large frame and thus cannot send out a pause frame
- Upon reception of the pause request the far-end has just started transmission of a large frame
- The eTSEC receives a burst of short frames with minimum inter-frame-gap (96bit times for ethernet)

Once the user has determined the worst case scenario for their application, they program the required free BD threshold into the eTSEC (via RQPRM[PBTHR]). Since different BD rings may have different sizes and expected packet arrival rates, a separate threshold is provided for each active ring. It is recommended that a threshold of at least three BDs is the practical minimum for gigabit ethernet links.

For the Rx descriptor controller to determine the number of free BDs remaining in the ring, it needs to know the following:

1. The location of the current BD being used by hardware
2. The location of the last BD that was released (freed) by software
3. The length of the Rx BD ring.

For each active ring, the current BD pointer (RBPTR<sub>n</sub>) is maintained by the eTSEC. Software will know both the size of the Rx ring and the location of the last freed BD. By providing the eTSEC with those values (via RQPRM[LEN] and RFBPTR respectively) the eTSEC will always know how many receive buffers are available to be consumed by incoming data.

The number of guaranteed free BDs in the ring is then determined by:

When  $RFBPTR_n < RBPTR_n$

$$\text{FreeBDs} = RQPRM_n[\text{LEN}] - RBPTR_n + RFBPTR_n$$

When  $RFBPTR_n > RBPTR_n$

$$\text{FreeBDs} = RFBPTR_n - RBPTR_n$$

When  $RBPTR_n = RFBPTR_n$  the number of free BDs in the ring is either one (since RFBPTR<sub>n</sub> points to a free BD) or equal to the ring length. Since the BD pointed to by RBPTR<sub>n</sub> may be either in use or about to be used, it is not considered in the free BD count. To resolve the case where the two pointers collide, the following logic applies:

If RBASE<sub>n</sub> was updated and thus initializes both RBPTR<sub>n</sub> and RFBPTR<sub>n</sub>, the ring is deemed empty.

If RFBPTR<sub>n</sub> is updated by a software write and matches RBPTR<sub>n</sub>, the ring is deemed empty.

If HW updates RBPTR<sub>n</sub> and the result matches RFBPTR<sub>n</sub>, the ring is deemed to have one BD remaining. Upon writing this BD back to memory (indicating the buffer is occupied) the ring is deemed to be full.

**Important.** There is a possibility that if software is severely backlogged in updating RFBPTR<sub>n</sub>, the hardware could wrap around the ring entirely, consume exactly the remaining number of BDs and not halt with a BSY error. If software then increments RFBPTR<sub>n</sub> to the next address (thereby equalling RBPTR<sub>n</sub>), the hardware will assume the ring is now empty (when in fact there is only a single BD freed up). This will

result in the hardware failing to maintain back pressure on the far end. Upon software incrementing  $RFBPTR_n$  a subsequent time, the wrap condition will be successfully detected and hardware will recognize a nearly full ring (rather than a nearly empty one). Since software can increment  $RFBPTR_n$  by any amount, it is not possible for hardware to determine in this case whether the user has cleared the entire ring or just one BD. Users can eliminate the possibility of this condition occurring by ensuring that  $RFBPTR_n$  is incremented by at least two BDs each time (i.e. clear at least two buffers whenever the RxBD unload routine is called).

Once the eTSEC determines that this threshold has been reached, back pressure will be applied accordingly. The type of back pressure that is applied will vary according to the physical interface that is used.

- **Half duplex Ethernet:** No support in this mode.
- **Full duplex Ethernet:** An IEEE 802.3 PAUSE frame (see sect. 15.6.3.9/15-151) will be issued as if the TCTRL[TFC\_PAUSE] bit was set. An internal counter will track the time the far end controller is expected to remain in pause (based on the setting of PTV[PT]). When that counter reaches half the value of PTV[PT], the eTSEC will reissue a pause frame if the free BD calculation for any ring is below the threshold for that ring. For example, if PTV[PT] is set to 10 quanta, a pause frame will be re-issued when five quanta have elapsed if the free BD threshold is still not met. A practical minimum for PTV[PT] of 4 quanta is recommended.
- **FIFO packet interface:** Link layer flow control will be asserted via use of the RFC signal (CRS pin). Flow control will be asserted for the entire time that free BD threshold is not met. The same mechanism is used for both GMII-style and encoded packet modes.

## 15.6.6.2 Software Use of Hardware-Initiated Back Pressure

### 15.6.6.2.1 Initialization

Software configures  $RBASE_n$  and  $RQPRM_n[LEN]$  according to the parameters for that ring. Then the number of free BDs that are required to prevent the eTSEC from automatically asserting flow control are programmed in  $RQPRM_n[FBTHR]$ . The receiver is then enabled.

Note: the act of programming  $RBASE_n$  will initialize  $RFBPTR_n$  to the start of the of the ring. When the ring is in this initial empty state, there is no concept of a last freed BD. In this case, the calculated number of free BDs is the size of the ring. Since the BD that the hardware is currently pointing to is to be considered in-use, the free BD count is actually one higher than the total available. As soon as the hardware consumes a BD (by writing it back to memory),  $RBPTR_n$  will advance and the free BD count will

### 15.6.6.2.2 Operation

As software frees BDs from the ring, it writes the physical address of the BD just freed to  $RFBPTR_n$ . The eTSEC will assert flow control if the distance (using modulo arithmetic) between  $RBPTR_n$  and  $RFBPTR_n$  is  $< RQPRM_n[FBTHR]$ . In multi-ring operation, if the free BD count of **any** active ring drops below the threshold for that ring, flow control will be asserted. Once enough BDs are freed for **all** active rings to meet their respective free BD thresholds, application of back pressure will cease.

Note: The eTSEC will not issue an exit pause frame (i.e. pause frame with PTV of 0x0000) once all active rings have sufficient BDs. Instead, it will wait for the far-end pause timer to expire and start re-transmission.

## 15.6.7 Buffer Descriptors

The eTSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse across the PowerQUICC network processor family. Drawing from the MPC8260 FEC BD programming model, the eTSEC descriptor base registers point to the beginning of BD rings. The eTSEC BD also expands upon the MPC8260 BD model to accommodate the eTSEC's unique features. However, the 8-byte data BD format is designed to be compatible with the existing MPC8260 BD model.

### 15.6.7.1 Data Buffer Descriptors

Data buffers are used in the transmission and reception of Ethernet frames (see [Figure 15-134](#)). Data BDs encapsulate all information necessary for the eTSEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of pre-fetching, a minimum of four buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings. Transmit rings are limited to a maximum size of 65536 BDs due to BD and frame data prefetching. Software also must have the data pointer pointing to a legal memory location. Within the status field, there exists an ownership bit which defines the current state of the buffer (pointed to by the data pointer). Other bits in the status field of the buffer descriptor are used to communicate status/control information between the eTSEC and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see [Figure 15-135](#)), all BDs must reside sequentially in memory. The eTSEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the eTSEC to loop back to the beginning of the BD chain. Software must initialize the TBASE and RBASE registers that point to the beginning transmit and receive BDs for eTSEC.

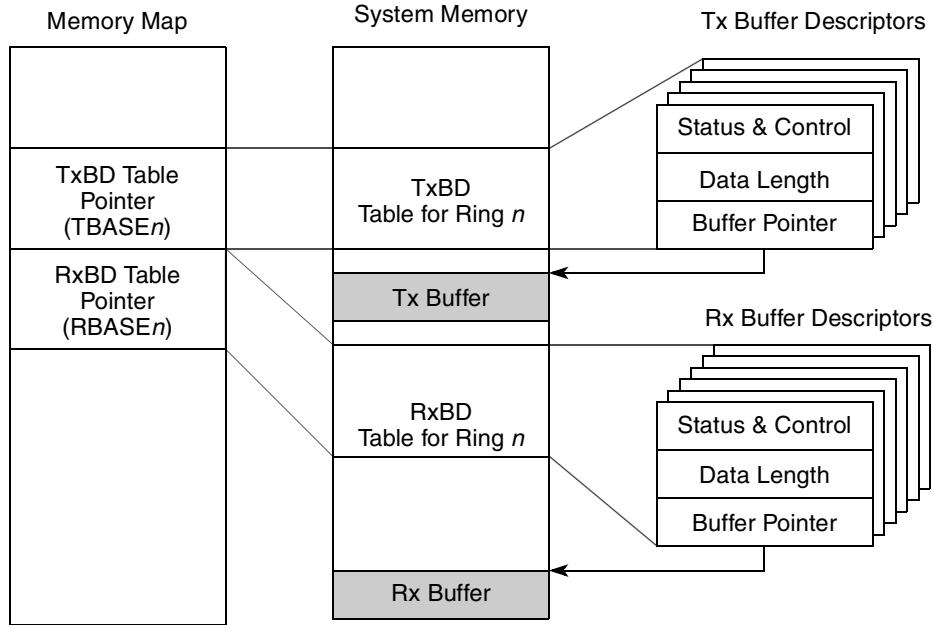


Figure 15-134. Example of eTSEC Memory Structure for BDs

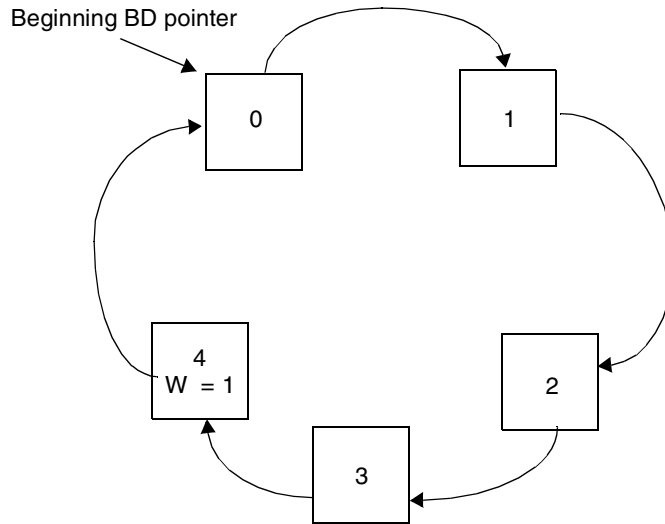


Figure 15-135. Buffer Descriptor Ring

### 15.6.7.2 Transmit Data Buffer Descriptors (TxBD)

Data is presented to the eTSEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, I, L, TC, PRE, HFE, CF, and TOE bits and the length (in bytes) in the first word, and the buffer pointer in the second word. Unused fields or fields written by the eTSEC must be initialized to zero. For transmission over the FIFO interface the Ethernet specific bits (PRE, DEF, HFE, LC, CF, RL and RC) have no meaning.

The eTSEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block.

Software must expect eTSEC to prefetch multiple TxBDs, and for TCP/IP checksumming an entire frame must be read from memory before a checksum can be computed. Accordingly, the R bit of the first TxBD in a frame must not be set until at least one entire frame can be fetched from this TxBD onwards. If eTSEC prefetches TxBDs and fails to reach a last TxBD (with bit L set), it will halt further transmission from the current TxBD ring and report an underrun error as IEVENT[XFUN]; this indicates that an incomplete frame was fetched, but remained unprocessed. The relevant TBPTR register points to the next unread TxBD following the error.

Figure 15-136 defines the TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	PAD/CRC	W	I	L	TC	PRE/DEF	0	HFE/LC	CF/RL	RC			TOE/UN	TR	
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

**Figure 15-136. Transmit Buffer Descriptor**

The TxBD definition is interpreted by eTSEC hardware as if TxBDs mapped to C data structures in the manner illustrated by Figure 15-137.

```

typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct txbd_struct {
    uint_16 flags;
    uint_16 length;
    uint_32 bufptr;
} txbd;

```

**Figure 15-137. Mapping of TxBDs to a C Data Structure**

The TxBD fields are detailed in [Table 15-146](#).

**Table 15-146. Transmit Data Buffer Descriptor (TxBD) Field Descriptions**

Offset	Bits	Name	Description
0–1	0	R	Ready, written by eTSEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The eTSEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	PAD/CRC	Padding for frames. (Valid only while it is set in the first BD and MACCFG2[PAD enable] is cleared). If MACCFG2[PAD enable] is sector if the eTSEC is operating in FIFO mode (ECTRL[FIFM] is set), this bit is ignored. 0 Do not add padding to short frames. 1 Add PAD to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, the eTSEC PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
	4	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.

Table 15-146. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	5	TC	<p>Tx CRC. Written by user. (Valid only while it is set in first BD and TxBD[PAD/CRC] is cleared and MACCFG2[PAD/CRC enable] is cleared and MACCFG2[CRC enable] is cleared.) If MACCFG2[PAD/CRC enable] is set or MACCFG2[CRC enable] is set, this bit is ignored in ethernet modes.</p> <p>If FIFOCFG[CRCAPP] is set, this bit is ignored in FIFO modes</p> <p>0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set.</p> <p>1 Transmit the CRC sequence after the last data byte.</p>
	6	PRE	<p>Transmit user-defined Ethernet preamble. Written by user. Valid only if set in the first BD of a frame, and MACCFG2[PreAm TxEN] is set.</p> <p>0 This frame does not contain Ethernet preamble bytes for transmission.</p> <p>1 This frame includes a user-defined Ethernet preamble sequence prior to the destination address in the data buffer.</p>
		DEF	<p>Defer indication. The eTSEC updates this bit after transmitting a frame (TxBD[L] is set)</p> <p>0 This frame was not deferred.</p> <p>1 This frame did not have a collision before it was sent but it was sent late because of deferring</p>
	7	—	Reserved
	8	HFE	<p>Huge frame enable. Written by user. Valid only if set in the first BD of a frame and MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored.</p> <p>0 Truncate transmit frame if its length is greater than the MAC's maximum frame length.</p> <p>1 Allow large frames to be transmitted without truncation.</p>
		LC	<p>Late collision. Written by the eTSEC.</p> <p>0 No late collision.</p> <p>1 A collision occurred after 64 bytes are sent. The eTSEC terminates the transmission and updates LC.</p>
	9	CF	<p>Control Frame. Written by user. Valid only if set in the first BD of a frame.</p> <p>0 Regular frame; transmission will be deferred when eTSEC is in PAUSE.</p> <p>1 Control frame; transmission will start even if eTSEC is in PAUSE.</p>
		RL	<p>Retransmission Limit. Written by the eTSEC.</p> <p>0 Transmission before maximum retry limit is hit.</p> <p>1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The eTSEC terminates the transmission and updates RL.</p>
	10–13	RC	<p>Retry Count. Written by the eTSEC.</p> <p>0 The frame is sent correctly the first time.</p> <p>x One or more attempts where needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.</p>

**Table 15-146. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)**

Offset	Bits	Name	Description
0–1	14	UN	Underrun. Written by the eTSEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. This could also have occurred in relation to a bus error causing IEVENT[EBERR]. The eTSEC terminates the transmission and updates UN.
		TOE	TCP/IP off-load enable. Written by user. Valid only if set in the first BD of a frame. 0 No TCP/IP off-load acceleration is applied to the frame prior to transmission. 1 eTSEC looks for a TOE Frame Control Block preceding the frame, and applies TCP/IP off-load acceleration as controlled by the FCB.
	15	TR	Truncation. Written by the eTSEC. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for a frame (a frame length greater than or equal to the value set in the maximum frame length register is encountered, the HFE bit in the BD is cleared, and MACCFG2[Huge Frame] is cleared). The frame is sent truncated.
2–3	0–15	Data Length	Data length is the number of octets the eTSEC should transmit from this BD's data buffer. It is never modified by the eTSEC. This field must be greater than zero, as zero indicates a BD not ready.
4–7	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

### 15.6.7.3 Receive Buffer Descriptors (RxBd)

In the RxBd the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the eTSEC modifies the E, L, F, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first word of the buffer descriptor are only modified by the eTSEC if the L (last BD in frame) bit is set. The first word of the RxBd contains control and status bits. Its formats are detailed below.

The number of buffer descriptors in a ring is set by using the W bit to indicate that the next buffer wraps back to the beginning of the ring. See [Section 15.5.3.5.5, “Maximum Frame Length Register \(MAXFRM\),”](#) for information on setting the size of the buffer ring.

[Figure 15-138](#) defines the RxBd.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	I	L	F	0	M	BC	MC	LG	NO	SH	CR	OV	TR
Offset + 2	DATA LENGTH															
Offset + 4	RX DATA BUFFER POINTER															
Offset + 6																

**Figure 15-138. Receive Buffer Descriptor**

The RxBd definition is interpreted by eTSEC hardware as if RxBds mapped to C data structures in the manner illustrated by [Figure 15-139](#).



```

typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct rxbd_struct {
    uint_16 flags;
    uint_16 length;
    uint_32 bufptr;
} rxbd;

```

**Figure 15-139. Mapping of RxBDs to a C Data Structure**

Table 15-147 describes the fields of the RxBD.

**Table 15-147. Receive Buffer Descriptor Field Descriptions**

Offset	Bits	Name	Description
0-1	0	E	Empty, written by the eTSEC (when cleared) and by the user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
	1	RO1	Receive software ownership bit. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
	2	W	Wrap, written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
	3	I	Interrupt, written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] or IMASK[RXFEN]). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
	4	L	Last in frame, written by the eTSEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
	5	F	First in frame, written by the eTSEC. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	—	Reserved
	7	M	Miss, written by the eTSEC. (This bit is valid only if the L-bit is set and eTSEC is in promiscuous mode.) This bit is set by the eTSEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition; thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.

Table 15-147. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	8	BC	Broadcast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
	9	MC	Multicast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.
	10	LG	Rx frame length violation, written by the eTSEC (only valid if L is set). A frame length greater than or equal to the maximum frame length was recognized; in this case LG is set regardless of the setting of MACCFG2[Huge Frame]. If MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the maximum frame length register. This bit is valid only if the L bit is set.
	11	NO	Rx non-octet aligned frame, written by the eTSEC (only valid if L is set). A frame that contained a number of bits not divisible by eight was received.
	12	SH	Short frame, written by the eTSEC (only valid if L is set). A frame length less than the minimum 64B that is defined for ethernet. was recognized, provided RCTRL[RSF] is set.
	13	CR	Rx CRC error, written by the eTSEC (only valid if L is set). This frame contains a CRC error and is an integral number of octets in length. This bit is also set if a receive code group error is detected.
	14	OV	Overflow, written by the eTSEC (only valid if L is set). A receive FIFO overflow occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR and TR lose their normal meaning and are zero.
	15	TR	Truncation, written by the eTSEC (only valid if L is set). Set if the receive frame is truncated. This can happen if a frame length greater than the maximum frame length is received and MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect. This bit is not set when in FIFO mode as truncation cannot occur.
2–3	0–15	Data Length	Data length, written by the eTSEC. Data length is the number of octets written by the eTSEC into this BD's data buffer if L is cleared (the value is equal to MRBLR), or, if L is set, the length of the frame including CRC, FCB (if RCTRL[PRSDEP > 00]) and any padding (RCTRL[PAL]).
4–7	0–31	RX Data Buffer Pointer	Receive buffer pointer, written by the user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 8-byte aligned. For best performance, use 64-byte aligned receive buffer pointer addresses. The buffer must reside in memory external to the eTSEC.

## 15.7 Initialization/Application Information

### 15.7.1 Interface Mode Configuration

This section describes how to configure the eTSEC in different supported interface modes. These include:

- MII
- RMII
- GMII

- RGMII
- TBI
- RTBI
- 8-bit FIFO

The pinout, the data registers that must be initialized, as well as speed selection options are described.

### 15.7.1.1 MII Interface Mode

Table 15-148 describes the signal configurations required for MII interface mode.

**Table 15-148. MII Interface Mode Signal Configuration**

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RX_DV	I	1

**Table 15-148. MII Interface Mode Signal Configuration (continued)**

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
RX_ER	I	1	RX_ER	I	1
COL	I	1	COL	I	1
CRS	I	1	CRS	I	1
<b>Sum</b>		25	<b>Sum</b>		16

Table 15-149 describes the shared signals of the MII interface.

**Table 15-149. Shared MII Signals**

eTSEC Signals	I/O	No. of Signals	MII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	not used	I	0	Reference clock
<b>Sum</b>			<b>Sum</b>			

Table 15-150 describes the register initializations required to configure the eTSEC in MII mode.

**Table 15-150. MII Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half duplex operation. Set I/F Mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.

**Table 15-150. MII Mode Register Initialization Steps (continued)**

Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0000_1100]
Perform an MII Mgmt write cycle to the external PHY Writing to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00.

**Table 15-150. MII Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle of Status Register.                  Clear MIIMCOM[Read Cycle].                  Set MIIMCOM[Read Cycle].                  (Uses the PHY address (0) and Register address (1) placed in MIIMADD register),                  When MIIMIND[BUSY]=0,                  read the MIIMSTAT register and check bit 10 (AN Done and Link is up)                  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0100]                  Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>
<p>Check auto-negotiation attributes in the PHY as necessary.</p>
<p>Clear IEVENT register,                  IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional)                  IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize MACnADDR1/2 (Optional)                  MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDRn (Optional)                  GADDRn[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional)                  RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize DMACTRL (Optional)                  DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize (Empty) Transmit Descriptor ring and fill buffers with Data                  Initialize TBASE0–TBASE7,                  TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Initialize (Empty) Receive Descriptor ring and fill with empty buffers                  Initialize RBASE0–RBASE7,                  RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Enable Transmit Queues                  Initialize TQUEUE</p>
<p>Enable Receive Queues                  Initialize RQUEUE</p>
<p>Enable Rx and Tx,                  MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]</p>

### 15.7.1.2 GMII Interface Mode

Table 15-151 describes the signal configurations required for GMII interface mode.

**Table 15-151. GMII Interface Mode Signal Configuration**

eTSEC Signal s			GMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 3.3		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	TxD[4]	O	1
TxD[5]	O	1	TxD[5]	O	1
TxD[6]	O	1	TxD[6]	O	1
TxD[7]	O	1	TxD[7]	O	1
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	RxD[4]	I	1
RxD[5]	I	1	RxD[5]	I	1
RxD[6]	I	1	RxD[6]	I	1
RxD[7]	I	1	RxD[7]	I	1
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
<b>Sum</b>		25	<b>Sum</b>		23

Table 15-152 describes the shared signals of the GMII interface.

**Table 15-152. Shared GMII Signals**

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
<b>Sum</b>			<b>Sum</b>			

Table 15-153 describes the register initializations required to configure the eTSEC in GMII mode.

**Table 15-153. GMII Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for GMII, Full duplex operation. Set I/F Mode bit. MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (This example has Full Duplex = 1, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface, MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY), MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]



**Table 15-153. GMII Mode Register Initialization Steps (continued)**

<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed</p>
<p>Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection  MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]</p>
<p>Perform an MII Mgmt write cycle to the external PHY.  Write to MII Mgmt Control with 16-bit data intended for the external PHY register,  MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection,  MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Perform an MII Mgmt write cycle to the external PHY.  Write to MII Mgmt Control with 16-bit data intended for the external PHY register,  MIIMCON[0000_0000_0000_0000_000u_00u1_0100_0000]  where u is user defined based on desired configuration.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation.  Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001]  The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00</p>
<p>Perform an MII Mgmt read cycle of Status Register.  Clear MIIMCOM[Read Cycle].  Set MIIMCOM[Read Cycle].  (Uses the PHY address (0) and Register address (1) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  Read the MIIMSTAT register and check bit 10 (AN Done and Link is up),  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0100]  Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>
<p>Check auto-negotiation attributes in the PHY as necessary.</p>
<p>Clear IEVENT register,  IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional)  IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize MACnADDR1/2 (Optional)  MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDR<sub>n</sub> (Optional)  GADDR<sub>n</sub>[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional)  RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>

**Table 15-153. GMII Mode Register Initialization Steps (continued)**

Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

### 15.7.1.3 TBI Interface Mode

Table 15-154 describes the signal configurations required for TBI interface mode.

**Table 15-154. TBI Interface Mode Signal Configuration**

eTSEC Signals			TBI Interface		
			Frequency [MHz] 62.5		
			Voltage [V] 3.3		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TCG[4]	O	1
TxD[5]	O	1	TCG[5]	O	1
TxD[6]	O	1	TCG[6]	O	1
TxD[7]	O	1	TCG[7]	O	1
TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RCG[4]	I	1
RxD[5]	I	1	RCG[5]	I	1
RxD[6]	I	1	RCG[6]	I	1
RxD[7]	I	1	RCG[7]	I	1
RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RCG[9]	I	1
COL	I	1	not used		
CRS	I	1	SDET	I	1
<b>Sum</b>		25	<b>Sum</b>		24

Table 15-155 describes the shared signals for the TBI interface.

**Table 15-155. Shared TBI Signals**

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
<b>Sum</b>			<b>Sum</b>			

Table 15-156 describes the register initializations required to configure the eTSEC in TBI mode.

**Table 15-156. TBI Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.

**Table 15-156. TBI Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle to verify state of TBI Control Register(Optional)  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the TBI address and Register address placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MIIMSTAT and look for AN Enable and other bit information.</p>
<p>Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100]  The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)</p>
<p>Perform an MII Mgmt write cycle to TBI.  Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register,  MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000]  This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000]  the Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)</p>
<p>Perform an MII Mgmt write cycle to TBI.  Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register,  MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000]  This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Check to see if PHY has completed Auto-Negotiation.  Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001]  The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.</p>
<p>Perform an MII Mgmt read cycle of Status Register.  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (2) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MIIMSTAT register and check bit 10 (AN Done)  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0000]  Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>

**Table 15-156. TBI Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle of AN Expansion Register.  Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd)  MII Mgmt AN Expansion ---&gt; [0000_0000_0000_0000_0000_0000_0000_0110]</p>
<p>Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional)  Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex)  MII Mgmt AN Link Partner Base Page Ability ---&gt; [0000_0000_0000_0000_0000_000x_x110_0000]</p>
<p>Clear IEVENT register,  IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional)  IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize MACnADDR1/2 (Optional)  MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDR<sub>n</sub> (Optional)  GADDR<sub>n</sub>[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional)  RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize DMACTRL (Optional)  DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize (Empty) Transmit Descriptor ring and fill buffers with Data  Initialize TBASE0–TBASE7,  TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Initialize (Empty) Receive Descriptor ring and fill with empty buffers  Initialize RBASE0–RBASE7,  RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Enable Transmit Queues  Initialize TQUEUE</p>
<p>Enable Receive Queues  Initialize RQUEUE</p>
<p>Enable Rx and Tx,  MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]</p>

### 15.7.1.4 RGMII Interface Mode

Table 15-157 shows the signals configurations required for RGMII interface mode.

**Table 15-157. RGMII Interface Mode Signal Configuration**

eTSEC Signals			RGMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
<b>Sum</b>		25	<b>Sum</b>		12

Table 15-158 describes the shared signals for the RGMII interface.

**Table 15-158. Shared RGMII Signals**

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
<b>Sum</b>			<b>Sum</b>			

Table 15-159 describes the register initializations required to configure the eTSEC in RGMII mode.

**Table 15-159. RGMII Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has RGMII 10Mbps mode, Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that EC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)



**Table 15-159. RGII Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.</p>
<p>Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.</p>
<p>Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)</p>
<p>Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.</p>
<p>Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.</p>
<p>Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.</p>
<p>Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>
<p>Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---&gt; [0000_0000_0000_0000_0000_0000_0000_0110]</p>
<p>Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---&gt; [0000_0000_0000_0000_0000_000x_1x10_0000]</p>

**Table 15-159. RGMII Mode Register Initialization Steps (continued)**

<p>Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDR<sub>n</sub> (Optional) GADDR<sub>n</sub>[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Enable Transmit Queues Initialize TQUEUE</p>
<p>Enable Receive Queues Initialize RQUEUE</p>
<p>Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]</p>

### 15.7.1.5 RMII Interface Mode

Table 15-160 shows the signals configurations required for RMII interface mode.

**Table 15-160. RMII Interface Mode Signal Configuration**

eTSEC Signals			RMII Interface		
			Frequency [MHz] 50		
			Voltage [V] 3.3		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	REF_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	leave unconnected		
TxD[3]	O	1	leave unconnected		
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	leave unconnected		
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	not used		
RxD[3]	I	1	not used		
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	CRS_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
<b>Sum</b>		25	<b>Sum</b>		8

Table 15-161 describes the shared signals for the RMII interface.

**Table 15-161. Shared RMII Signals**

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
TX_CLK	I	1	REF_CLK	I	1	Reference clock
<b>Sum</b>		<b>3</b>	<b>Sum</b>		<b>3</b>	

Table 15-162 describes the register initializations required to configure the eTSEC in RMII mode.

**Table 15-162. RMII Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0001_0000] (Used to setup Reduced-Pin mode = 1, and TBIM = 0, statistics enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543 for example
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543 for example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_1101] set system clock divide by 14 for example to insure that MDC clock speed = 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.

**Table 15-162. RMII Mode Register Initialization Steps (continued)**

<p>Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000]  The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)</p>
<p>Perform an MII Mgmt write cycle to the external PHY.  Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register,  MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000]  This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Check to see if PHY has completed Auto-Negotiation.  Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001]  The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.</p>
<p>Perform an MII Mgmt read cycle of Status Register.  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (1) placed in MIIMADD register)  When MIIMIND[BUSY]=0,  read the MIIMSTAT register and check bit 10. (AN Done)  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0000]  Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>
<p>Perform an MII Mgmt read cycle of AN Expansion Register.  Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register)  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd)  MII Mgmt AN Expansion ---&gt; [0000_0000_0000_0000_0000_0000_0000_0110]</p>
<p>Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional)  Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register)  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex)  MII Mgmt AN Link Partner Base Page Ability ---&gt; [0000_0000_0000_0000_0000_000x_x110_0000]</p>
<p>Setting up the MII Mgmt for a write cycle to TBI MII Mgmt register (write the TBI's address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_1011]  the TBI control register is at offset address 0x11 from TBIPA</p>
<p>Perform an MII Mgmt write cycle  Writing to MII Mgmt Control with 16-bit data intended for TBI's MII Mgmt control register (TBI control),  MIIMCON[0000_0000_0000_0000_0000_0010_0001_0000]  This configures the TBI control to GMII mode and AN sense</p>
<p>Check to see if MII Mgmt write is complete  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicate that the write cycle was completed</p>

**Table 15-162. RMII Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle (Optional)  Set MIIMCOM[Read Cycle]  (Uses the TBI address and Register address placed in MIIMADD register),  read the MIIMSTAT register and verify that  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0010_0001_0000]</p>
<p>Check to see if PHY has completed Auto-Negotiation  Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address),  MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010]  the PHY Status control register is at address 0x2 and lets say the PHY Address is 0x2</p>
<p>Perform an MII Mgmt read cycle of Status Register  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (2) placed in MIIMADD register),  read the MIIMSTAT register and check bit 10 (AN Done)  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0000]  other information about the link is also returned (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>
<p>Perform an MII Mgmt read cycle of AN Expansion Register  MIIMADD[0000_0000_0000_0000_0000_0010_0000_0110]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (6) placed in MIIMADD register),  read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd)  MII Mgmt AN Expansion ---&gt; [0000_0000_0000_0000_0000_0000_0000_0110]</p>
<p>Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register (Optional)  MIIMADD[0000_0000_0000_0000_0000_0010_0000_0101]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (5) placed in MIIMADD register),  read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10 (Half and Full Duplex)  MII Mgmt AN Link Partner Base Page Ability ---&gt; [0000_0000_0000_0000_0000_000X_1110_0000]</p>
<p>Clear IEVENT register,  IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional)  IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDR<sub>n</sub> (Optional)  GADDR<sub>n</sub>[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional)  RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize DMACTRL (Optional)  DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize (Empty) Transmit Descriptor ring and fill buffers with Data  Initialize TBASE0–TBASE7,  TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Initialize (Empty) Receive Descriptor ring and fill with empty buffers  Initialize RBASE0–RBASE7,  RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Enable Transmit Queues  Initialize TQUEUE</p>
<p>Enable Receive Queues  Initialize RQUEUE</p>
<p>Enable Rx and Tx,  MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]</p>

### 15.7.1.6 RTBI Interface Mode

Table 15-163 describes the signal configurations required for RTBI interface mode.

**Table 15-163. RTBI Interface Mode Signal Configuration**

eTSEC Signal s			RTBI Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
<b>Sum</b>		25	<b>sum</b>		12

Table 15-164 describes the shared signals for the RTBI interface.

**Table 15-164. Shared RTBI Signals**

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
Sum			Sum			

Table 15-165 describes the register initializations required to configure the eTSEC in RTBI mode.

**Table 15-165. RTBI Mode Register Initialization Steps**

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that EC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.



**Table 15-165. RTBI Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle to verify state of TBI Control Register(Optional)  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the TBI address and Register address placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MIIMSTAT and look for AN Enable and other bit information.</p>
<p>Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100]  The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)</p>
<p>Perform an MII Mgmt write cycle to TBI.  Write to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register,  MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000]  This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000]  The Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)</p>
<p>Perform an MII Mgmt write cycle to TBI.  Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register,  MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000]  This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.</p>
<p>Check to see if MII Mgmt write is complete.  Read MII Mgmt Indicator register and check for Busy = 0,  MIIMIND ---&gt; [0000_0000_0000_0000_0000_0000_0000_0000]  This indicates that the write cycle was completed.</p>
<p>Check to see if PHY has completed Auto-Negotiation.  Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address),  MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001]  The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.</p>
<p>Perform an MII Mgmt read cycle of Status Register.  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (2) and Register address (2) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MIIMSTAT register and check bit 10 (AN Done)  MIIMSTAT ---&gt; [0000_0000_0000_0000_0000_0000_0010_0000]  Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)</p>

**Table 15-165. RTBI Mode Register Initialization Steps (continued)**

<p>Perform an MII Mgmt read cycle of AN Expansion Register.  Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd)  MII Mgmt AN Expansion ---&gt; [0000_0000_0000_0000_0000_0000_0000_0110]</p>
<p>Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional)  Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101]  Clear MIIMCOM[Read Cycle]  Set MIIMCOM[Read Cycle]  (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register),  When MIIMIND[BUSY]=0,  read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex)  MII Mgmt AN Link Partner Base Page Ability ---&gt; [0000_0000_0000_0000_0000_0000_00x_x110_0000]</p>
<p>Clear IEVENT register,  IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize IMASK (Optional)  IMASK[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize MACnADDR1/2 (Optional)  MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize GADDR<sub>n</sub> (Optional)  GADDR<sub>n</sub>[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize RCTRL (Optional)  RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize DMACTRL (Optional)  DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]</p>
<p>Initialize (Empty) Transmit Descriptor ring and fill buffers with Data  Initialize TBASE0–TBASE7,  TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Initialize (Empty) Receive Descriptor ring and fill with empty buffers  Initialize RBASE0–RBASE7,  RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]</p>
<p>Enable Transmit Queues  Initialize TQUEUE</p>
<p>Enable Receive Queues  Initialize RQUEUE</p>
<p>Enable Rx and Tx,  MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]</p>

### 15.7.1.7 8-Bit FIFO Mode

Table 15-166 describes the signal configurations required for 8-bit FIFO interface mode.

**Table 15-166. 8-Bit FIFO Interface Mode Signal Configurations**

eTSEC Signals			8-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 3.3/2.5		
Signals	I/O	No. of Signals	Signals	I/O	No. of Signals
TSEC <sub>n</sub> _GTX_CLK	O	1	FIFO <sub>n</sub> _GTX_CLK	O	1
TSEC <sub>n</sub> _TX_CLK	I	1	FIFO <sub>n</sub> _TX_CLK	I	1
TSEC <sub>n</sub> _TxD[0]	O	1	FIFO <sub>n</sub> _TxD[0]	O	1
TSEC <sub>n</sub> _TxD[1]	O	1	FIFO <sub>n</sub> _TxD[1]	O	1
TSEC <sub>n</sub> _TxD[2]	O	1	FIFO <sub>n</sub> _TxD[2]	O	1
TSEC <sub>n</sub> _TxD[3]	O	1	FIFO <sub>n</sub> _TxD[3]	O	1
TSEC <sub>n</sub> _TxD[4]	O	1	FIFO <sub>n</sub> _TxD[4]	O	1
TSEC <sub>n</sub> _TxD[5]	O	1	FIFO <sub>n</sub> _TxD[5]	O	1
TSEC <sub>n</sub> _TxD[6]	O	1	FIFO <sub>n</sub> _TxD[6]	O	1
TSEC <sub>n</sub> _TxD[7]	O	1	FIFO <sub>n</sub> _TxD[7]	O	1
TSEC <sub>n</sub> _TX_EN	O	1	FIFO <sub>n</sub> _TX_EN	O	1
TSEC <sub>n</sub> _TX_ER	O	1	FIFO <sub>n</sub> _TX_ER	O	1
TSEC <sub>n</sub> _RX_CLK	I	1	FIFO <sub>n</sub> _RX_CLK	I	1
TSEC <sub>n</sub> _RxD[0]	I	1	FIFO <sub>n</sub> _RxD[0]	I	1
TSEC <sub>n</sub> _RxD[1]	I	1	FIFO <sub>n</sub> _RxD[1]	I	1
TSEC <sub>n</sub> _RxD[2]	I	1	FIFO <sub>n</sub> _RxD[2]	I	1
TSEC <sub>n</sub> _RxD[3]	I	1	FIFO <sub>n</sub> _RxD[3]	I	1
TSEC <sub>n</sub> _RxD[4]	I	1	FIFO <sub>n</sub> _RxD[4]	I	1
TSEC <sub>n</sub> _RxD[5]	I	1	FIFO <sub>n</sub> _RxD[5]	I	1
TSEC <sub>n</sub> _RxD[6]	I	1	FIFO <sub>n</sub> _RxD[6]	I	1
TSEC <sub>n</sub> _RxD[7]	I	1	FIFO <sub>n</sub> _RxD[7]	I	1
TSEC <sub>n</sub> _RX_DV	I	1	FIFO <sub>n</sub> _RX_DV	I	1
TSEC <sub>n</sub> _RX_ER	I	1	FIFO <sub>n</sub> _RX_ER	I	1
TSEC <sub>n</sub> _COL	I	1	FIFO <sub>n</sub> _TX_FC	I	1
TSEC <sub>n</sub> _CRS	I/O	1	FIFO <sub>n</sub> _RX_FC	O	1
TSEC1_MDIO	I/O	1	leave unconnected		
TSEC1_MDC	O	1	leave unconnected		
<b>Sum</b>		<b>27</b>	<b>Sum</b>		<b>25</b>

Table 15-167 describes the register initializations required to configure the eTSEC in 8-bit FIFO mode.

**Table 15-167. 8-Bit FIFO Mode Register Initialization Steps**

Set FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_1100_0000_0000_0000] (Reset RX = 1, reset Tx = 1, Rx enable = 0, Tx enable = 0)
Clear FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_0000_0000_0000_1000] (Reset RX = 0, reset Tx = 0, Rx enable = 0, Tx enable = 0)
Ensure MACCFG2 is set to default values. MACCFG2[0000_0000_0000_0000_0111_0000_0000_0000]
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_1000_0000_0000_0000] (Used to set up FIFO mode = 1, and statistics enable = 0)
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) transmit descriptor ring and fill buffers with data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx over FIFO, FIFOCFG[0000_0000_0000_0000_0011_0000_1101_1000] (Rx enable = 1, Tx enable = 1, enable flow control and CRC, 8-bit mode)

# Chapter 16

## DMA Controller

This chapter describes the DMA controller of the MPC8533E.

### 16.1 Introduction

The DMA controller transfers blocks of data between PCI, PCI Express, the local bus controller (LBC) interface, and the local address space, independent of the e500 core or external hosts.

#### 16.1.1 Block Diagram

Figure 16-1 shows the block diagram of the DMA controller.

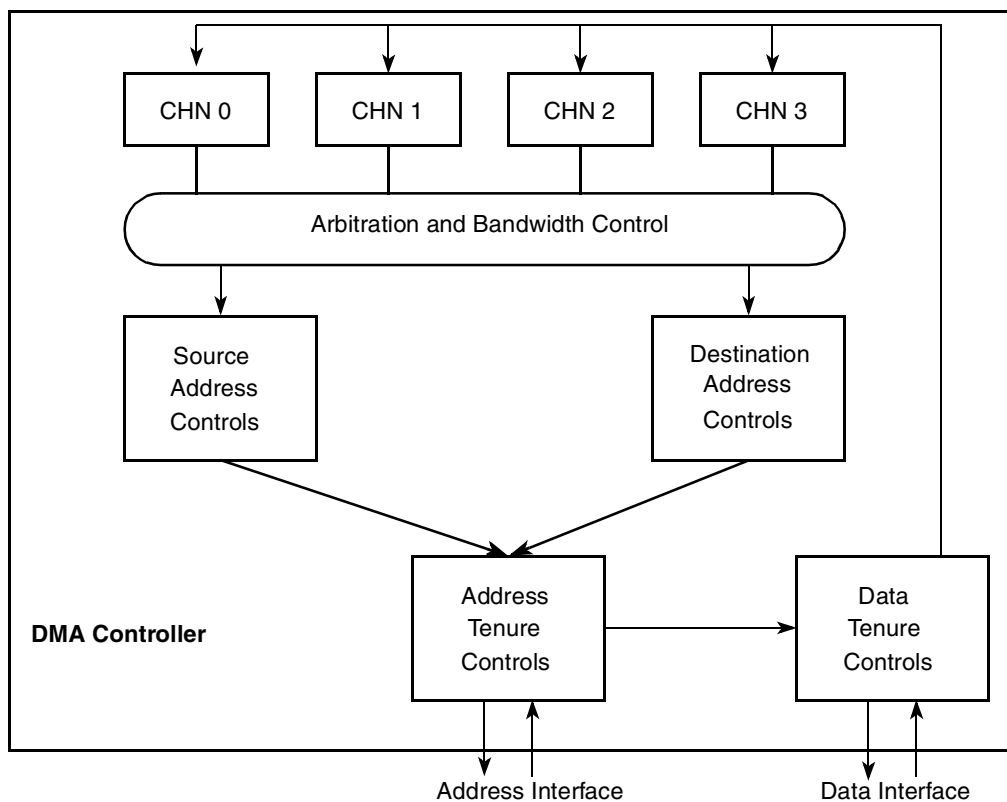


Figure 16-1. DMA Block Diagram

## 16.1.2 Overview

The DMA controller has four high-speed DMA channels. Both the e500 core and external devices can initiate DMA transfers. All channels are capable of complex data movement and advanced transaction chaining. [Figure 16-1](#) is a high-level block diagram of the DMA controller. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

## 16.1.3 Features

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Up to 256 bytes for DMA sub-block transfers to maximize performance
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using `DMA_DREQ`, `DMA_DACK`, and `DMA_DDONE`

## 16.1.4 Modes of Operation

The MPC8533E has two modes of operation: basic and extended. Basic mode is the DMA legacy mode. It does not support advanced features. Extended mode supports advanced features like striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. [Table 16-1](#) summarizes the relationship between the modes and the following features:

- Direct mode. No descriptors are involved. Software must initialize the required fields as described in [Table 16-1](#) before starting a transfer.
- Chaining mode. Software must initialize descriptors in memory and the required fields as described in [Table 16-1](#) before starting a transfer.
- Single-write start mode. The DMA process can be started by using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- External control capability. This allows an external agent to start, pause, and check the status of a DMA transfer which has already been initialized.

- Channel continue capability. The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability. The software can abort a previously initiated transfer by setting the bit  $MR_n[CA]$ . The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

**Table 16-1. Relationship of Modes and Features**

Mode	Mode with One Additional Feature	Mode with Two Additional Features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (extended chaining)	ExtCE (ExtC external control)
		ExtCS (ExtC single-write start)

Table 16-2 describes bit settings required for each DMA mode of operation.

**Table 16-2. DMA Mode Bit Settings**

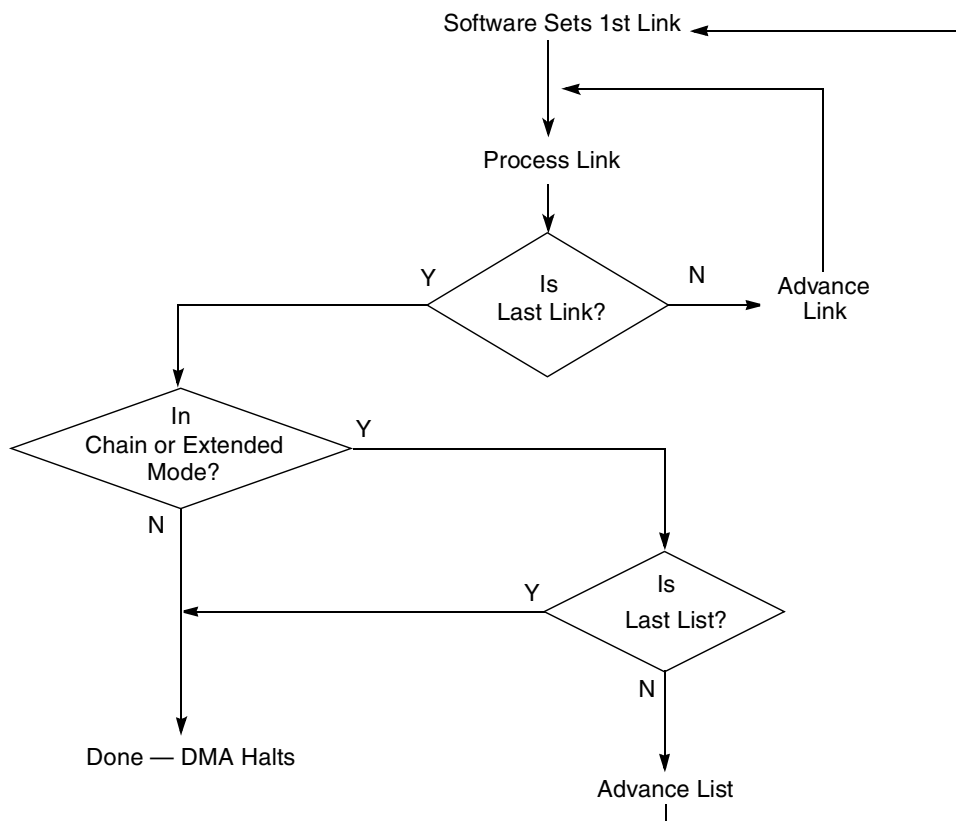
Modes with Features	$MR_n[XFE]$	$MR_n[CTM]$	$MR_n[SRW]$	$MR_n[CDSM/SWSM]$	$MR_n[EMS\_EN]$
Basic Direct Modes					
Basic direct	0	1	0	0	0
Basic direct external control	0	1	0	0	1
Basic direct single-write start	0	1	1	1 or 0	0
Basic Chaining Modes					
Basic chaining	0	0	Reserved	0	0
Basic chaining external control	0	0	Reserved	0	1
Basic chaining single-write start	0	0	Reserved	1	0
Extended Direct Modes					
Extended direct	1	1	0	0	0
Extended direct external control	1	1	0	0	1
Extended direct single-write start	1	1	1	1 or 0	0

**Table 16-2. DMA Mode Bit Settings (continued)**

Modes with Features	MRn[XFE]	MRn[CTM]	MRn[SRW]	MRn[CDSM/SWSM]	MRn[EMS_EN]
Extended Chaining Modes					
Extended chaining	1	0	Reserved	0	0
Extended chaining external control	1	0	Reserved	0	1
Extended chaining single-write start	1	0	Reserved	1	0

Refer to [Section 16.4, “Functional Description,”](#) for details on these modes.

Figure 16-2 shows the general DMA operational flow chart.



**Figure 16-2. DMA Operational Flow Chart**



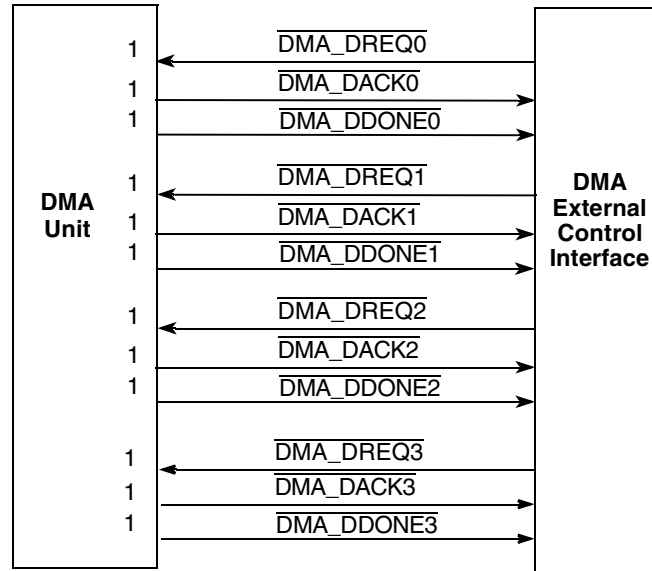
## 16.2 External Signal Description

This section describes the DMA signals.

### 16.2.1 Signal Overview

Figure 16-3 summarizes the DMA controller signals.

Figure 16-3. DMA Signal Summary



Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the device. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in [Section 19.4.1.11, “Alternate Function Signal Multiplex Control Register \(PMUXCR\).”](#)

## 16.2.2 Detailed Signal Descriptions

Table 16-3 describes the DMA signals.

**Table 16-3. DMA Signals—Detailed Signal Descriptions**

Signal	I/O	Description
$\overline{\text{DMA\_DREQ}}_n$ DMA request	I	DMA request. The DMA request signal indicates the start of a DMA transfer or a restart from a paused request. Assertion of $\overline{\text{DMA\_DREQ}}_n$ causes $\text{MR}_n[\text{CS}]$ to be set, thereby activating the corresponding DMA channel.
		<b>State Meaning</b> Asserted—Assertion of $\overline{\text{DMA\_DREQ}}_n$ while $\overline{\text{DMA\_DACK}}_n$ is negated causes a new transfer to start OR resumes a paused transfer if the $\text{EMP\_EN}$ bit is set. Assertion while $\overline{\text{DMA\_DACK}}_n$ is asserted results in an illegal condition. Negated—Negation while $\overline{\text{DMA\_DACK}}_n$ is asserted has no effect. Negation before the assertion of $\overline{\text{DMA\_DACK}}_n$ results in an illegal condition.
		<b>Timing</b> Assertion—Can be asserted asynchronously Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA\_DACK}}_n$
$\overline{\text{DMA\_DACK}}_n$	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress
		<b>State Meaning</b> Asserted—Indicates that a DMA transfer is currently in progress. Asserted after the assertion of $\overline{\text{DMA\_DREQ}}_n$ to indicate the start of a transfer Negated—Negated after finishing a complete transfer or after entering a paused state if $\text{MR}_n[\text{EMP\_EN}]$ is set
		<b>Timing</b> Assertion—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
$\overline{\text{DMA\_DDONE}}_n$	O	DMA done. Indicates that a DMA transfer is complete
		<b>State Meaning</b> Asserted—Indicates transfer completion. $\text{SR}_n[\text{CB}]$ is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated—Indicates that the current transfer is in process
		<b>Timing</b> Assertion—Always asserts asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}_n$ to indicate completion of a transfer. For a paused transfer, $\overline{\text{DMA\_DDONE}}_n$ is asserted asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}_n$ . Negation—Negated asynchronously after the assertion of $\overline{\text{DMA\_DREQ}}_n$ for the next transfer

## 16.3 Memory Map/Register Definition

This section provides a detailed description of all accessible DMA memory and registers. The descriptions include individual bit level descriptions and reset states of each register. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 16-4 lists the DMA registers and their offsets. Note that the full register address is comprised of the programmable CCSRBAR together with the fixed DMA block base address and offset listed in Table 16-4.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.

- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 16-4. DMA Register Summary**

Offset	Register	Access	Reset	Section/Page
<b>DMA Controller Block Base Address: 0x2_1000</b>				
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	<a href="#">16.3.1.1/16-10</a>
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	<a href="#">16.3.1.2/16-12</a>
0x108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	<a href="#">16.3.1.4/16-15</a>
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	<a href="#">16.3.1.5/16-16</a>
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	<a href="#">16.3.1.6/16-17</a>
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	<a href="#">16.3.1.7/16-18</a>
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	<a href="#">16.3.1.8/16-19</a>
0x124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	<a href="#">16.3.1.12/16-22</a>
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	<a href="#">16.3.1.13/16-23</a>
0x148–0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	<a href="#">16.3.1.1/16-10</a>
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	<a href="#">16.3.1.2/16-12</a>
0x188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	<a href="#">16.3.1.4/16-15</a>
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	<a href="#">16.3.1.5/16-16</a>
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	<a href="#">16.3.1.6/16-17</a>
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	<a href="#">16.3.1.7/16-18</a>

Table 16-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	16.3.1.8/16-19
0x1A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	16.3.1.9/16-19
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	16.3.1.9/16-19
0x1B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	16.3.1.10/16-21
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	16.3.1.10/16-21
0x1B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	16.3.1.11/16-22
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	16.3.1.11/16-22
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	16.3.1.12/16-22
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	16.3.1.13/16-23
0x1C8– 0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	16.3.1.1/16-10
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	16.3.1.2/16-12
0x208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	16.3.1.3/16-13
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	16.3.1.3/16-13
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	16.3.1.4/16-15
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	16.3.1.5/16-16
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	16.3.1.6/16-17
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	16.3.1.7/16-18
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	16.3.1.8/16-19
0x224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	16.3.1.9/16-19
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	16.3.1.9/16-19
0x230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	16.3.1.10/16-21
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	16.3.1.10/16-21
0x238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	16.3.1.11/16-22
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	16.3.1.11/16-22
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	16.3.1.12/16-22
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	16.3.1.13/16-23
0x248– 0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	16.3.1.1/16-10

Table 16-4. DMA Register Summary (continued)

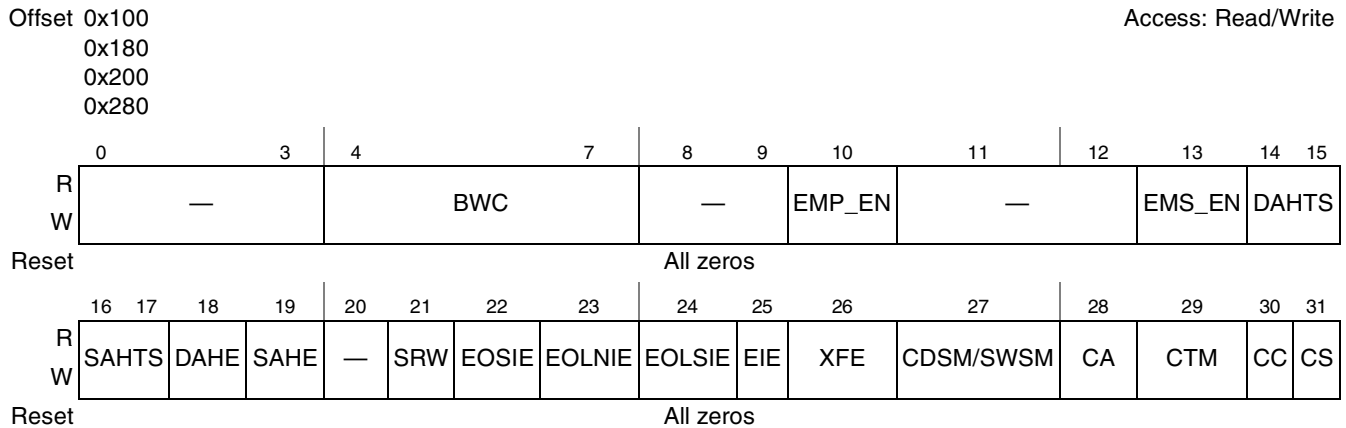
Offset	Register	Access	Reset	Section/Page
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	<a href="#">16.3.1.2/16-12</a>
0x288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	<a href="#">16.3.1.4/16-15</a>
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	<a href="#">16.3.1.5/16-16</a>
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	<a href="#">16.3.1.6/16-17</a>
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	<a href="#">16.3.1.7/16-18</a>
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	<a href="#">16.3.1.8/16-19</a>
0x2A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x2B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x2B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	<a href="#">16.3.1.12/16-22</a>
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	<a href="#">16.3.1.13/16-23</a>
0x2C8– 0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	<a href="#">16.3.1.14/16-24</a>

### 16.3.1 DMA Register Descriptions

The following sections describe the DMA registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register.

### 16.3.1.1 Mode Registers (MR<sub>n</sub>)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. Figure 16-4 describes the MR<sub>n</sub>.



**Figure 16-4. DMA Mode Registers (MR<sub>n</sub>)**

Table 16-5 describes the MR<sub>n</sub> fields.

**Table 16-5. MR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. If multiple channels are executing transfers concurrently the value of MR <sub>n</sub> [BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and switches to the next channel. If only one channel is executing transfers the value of MR <sub>n</sub> [BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of $\overline{\text{DREQ}}$ resumes channel operation. 0000 1 byte <span style="float: right;">0111 128 bytes</span> 0001 2 bytes <span style="float: right;">1000 256 bytes</span> 0010 4 bytes <span style="float: right;">1001 512 bytes</span> 0011 8 bytes <span style="float: right;">1010 1024 bytes</span> 0100 16 bytes <span style="float: right;">1011–1110 Reserved</span> 0101 32 bytes <span style="float: right;">1111 Disable bandwidth sharing to allow</span> 0110 64 bytes <span style="float: right;">uninterrupted transfers from each channel.</span>
8–9	—	Reserved
10	EMP_EN	External master pause enable. Valid only if MR <sub>n</sub> [EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MR <sub>n</sub> [BWC].
11–12	—	Reserved
13	EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MR <sub>n</sub> [CS].

Table 16-5. MR $n$  Field Descriptions (continued)

Bits	Name	Description
14–15	DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MR $n$ [DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MR $n$ [DAHTS] must be equal to or smaller than that assigned to MR $n$ [BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17	SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MR $n$ [SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MR $n$ [SAHTS] must be equal to or smaller than that assigned to MR $n$ [BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
18	DAHE	Destination address hold enable 0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MR $n$ [DAHTS]. Hardware only supports aligned transfers for this feature.
19	SAHE	Source address hold enable 0 Disable source address hold 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MR $n$ [SAHTS]. Hardware only supports aligned transfers for this feature.
20	—	Reserved
21	SRW	Single register write (Direct mode only; reserved for chaining mode.) 0 Normal operation 1 Enable a write to the source address register to simultaneously set MR $n$ [CS], starting a DMA transfer, when MR $n$ [CDSM/SWSM] is also set. Setting this bit and clearing CDSM/SWSM causes a write to the destination address register to simultaneously set MR $n$ [CS], starting a DMA transfer.
22	EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDAR $n$ [EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SR $n$ [EOSI] is set). This bit overrides the CLNDAR $n$ [EOSIE].
23	EOLNIE	End-of-links interrupt enable 0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDAR $n$ [EOLND] is set).
24	EOLSIE	End-of-lists interrupt enable 0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDAR $n$ [EOLND] and NLSDAR $n$ [EOLSD] are set).
25	EIE	Error interrupt enable 0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.

**Table 16-5. MRn Field Descriptions (continued)**

Bits	Name	Description
26	XFE	Extended features enable 0 Disable the new chaining features. 1 Enable the new chaining features.
27	CDSM/ SWSM	<ul style="list-style-type: none"> <li>In chaining mode: current descriptor start mode/single-write start mode                             <ul style="list-style-type: none"> <li>In basic mode (MRn[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MRn[CS], starting a DMA transfer.</li> <li>In extended chaining mode (MRn[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MRn[CS], starting a DMA transfer.</li> </ul> </li> <li>In direct mode: Setting this bit and MRn[SRW] causes a write to the source address register to simultaneously set MRn[CS], starting a DMA transfer. Clearing this bit and setting MRn[SRW] causes a write to the destination address register to simultaneously set MRn[CS], starting a DMA transfer. This bit must be cleared when MRn[SRW] is cleared.</li> </ul>
28	CA	Channel abort 0 No effect 1 Cause the current transfer to be aborted and SRn[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.
29	CTM	Channel transfer mode 0 Configure the channel in chaining mode. 1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.
30	CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31	CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. 0 Halt the DMA process if channel is busy (SRn[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

**16.3.1.2 Status Registers (SRn)**

The status registers, shown in [Figure 16-5](#), report various DMA conditions during and after a DMA transfer.



**Figure 16-5. Status Registers (SRn)**



Table 16-6 describes the bits of the  $SR_n$ .

**Table 16-6.  $SR_n$  Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24	TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See <a href="#">Section 16.4.3, “DMA Errors,”</a> for additional information.
25	—	Reserved
26	CH	Channel halted. Cleared automatically by hardware if $MR_n[CS]$ is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel ( $SR_n[CB]$ is cleared), this bit will remain 0. 1 DMA transfer was successfully halted by software and can be resumed.
27	PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28	EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if $MR_n[EOLSIE]$ is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29	CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 A DMA transfer is currently in progress.
30	EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if $MR_n[EOSIE]$ is set or if $CLNDAR_n[EOSIE]$ is set, this bit gets set and an interrupt is generated. In direct mode, if $MR_n[EOSIE]$ is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31	EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if $MR_n[EOLSIE]$ is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

### 16.3.1.3 Current Link Descriptor Address Registers ( $CLNDAR_n$ and $ECLNDAR_n$ )

Current link descriptor address registers contain the address of the current link descriptor. In basic chaining mode, shown in [Figure 16-6](#), software must initialize these registers to point to the first link descriptors in memory.

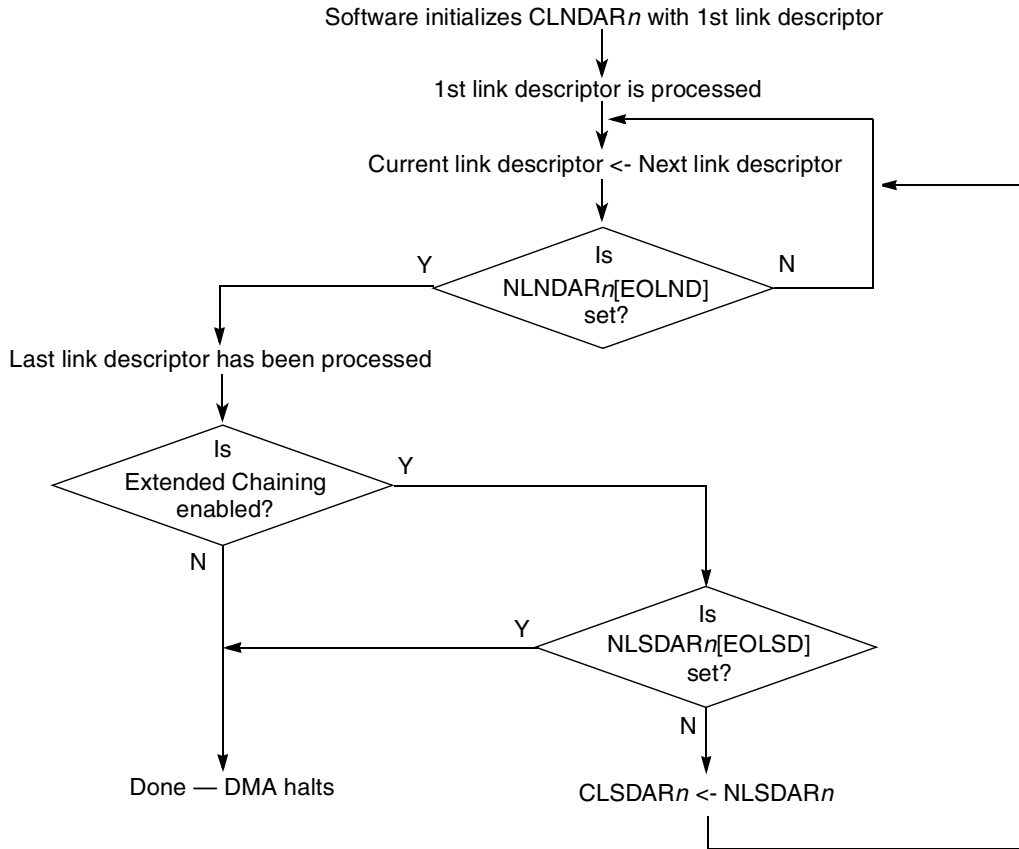


Figure 16-6. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address registers and  $NLNDAR_n[EOLND]$  in the next link descriptor address register is examined. If  $EOLND$  is zero, the DMA controller reads in the new current link descriptor for processing. If  $EOLND$  is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of  $NLSAR_n[EOLSD]$  in the next list descriptor address register. If  $EOLSD$  is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If  $EOLSD$  is set, all DMA transfers are complete and the DMA controller halts.

Figure 16-7 shows  $ECLNDAR_n$ .

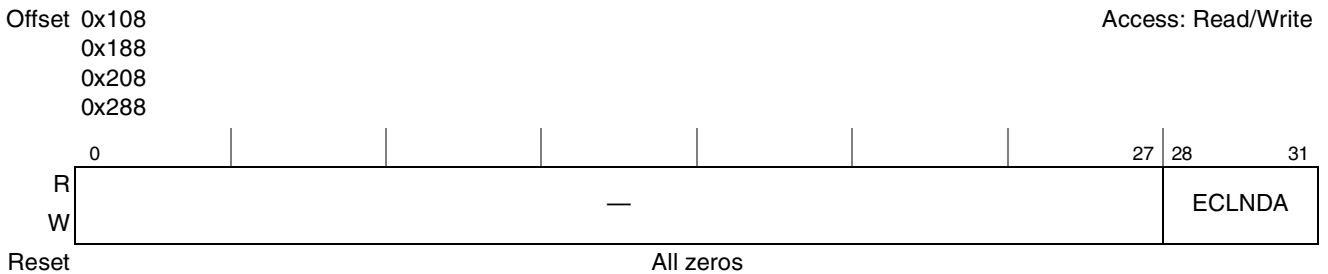


Figure 16-7. Extended Current Link Descriptor Address Registers ( $ECLNDAR_n$ )

Table 16-7. ECLNDAR<sub>n</sub> Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	ECLNDA	Current link descriptor extended address (upper 4 bits of 36-bit address)

Figure 16-8 shows CLNDAR<sub>n</sub>.

Figure 16-8. Current Link Descriptor Address Registers (CLNDAR<sub>n</sub>)

Table 16-8 describes the fields of the CLNDAR<sub>n</sub>.

Table 16-8. CLNDAR<sub>n</sub> Field Descriptions

Bits	Name	Description
0–26	CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary. (This is the lower portion of the 36-bit address formed by CLNDAR <sub>n</sub> [CLNDA] and ECLNDAR <sub>n</sub> [ECLNDA].)
27	—	Reserved
28	EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31	—	Reserved

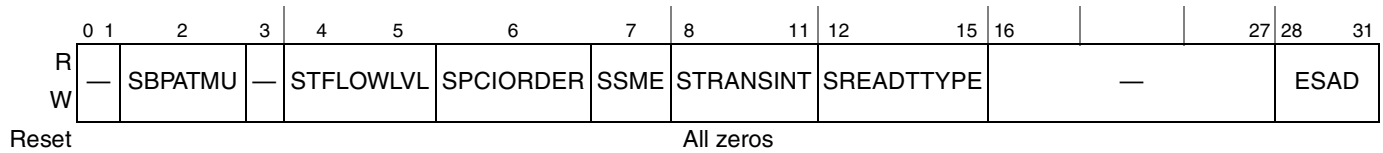
#### 16.3.1.4 Source Attributes Registers (SATR<sub>n</sub>)

The source attributes registers, shown in Figure 16-9, contain the transaction attributes to be used for the DMA operation. Stride mode is enabled by setting SATR<sub>n</sub>[SSME].

If SATR<sub>n</sub>[SBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in SATR<sub>n</sub>[SREADTTYPE] using the local address space category.

Offset 0x110  
 0x190  
 0x210  
 0x290

Access: Read/Write



**Figure 16-9. Source Attributes Registers (SATR<sub>n</sub>)**

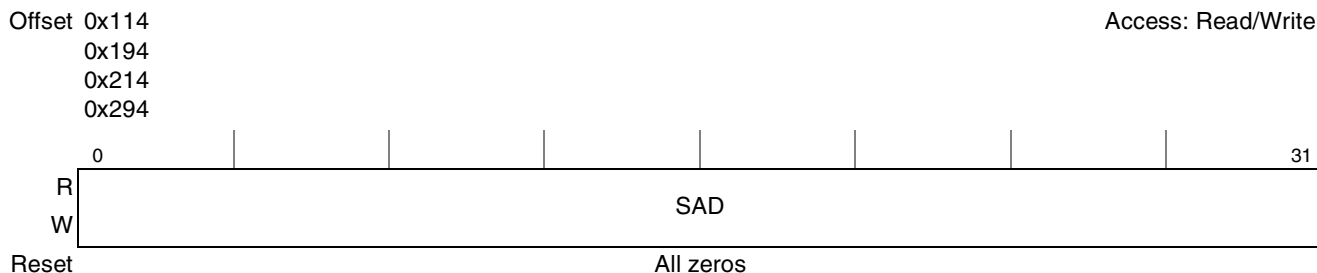
Table 16-9 describes the fields of the SATR<sub>n</sub>.

**Table 16-9. SATR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–6	—	Reserved
7	SSME	Source stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR <sub>n</sub> [XFE] is cleared). Striding on the source address can be accomplished by enabling SATR <sub>n</sub> [SSME] and setting the desired stride size and distance in the SSR <sub>n</sub> .
8–11	—	Reserved
12–15	SREADTTYPE	DMA source transaction type. Reserved values will result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space 0000–0001 Reserved 0011 Reserved 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0111 Read, unlock L2 cache line 1000–1111 Reserved
16–27	—	Reserved
28–31	ESAD	Extended source address. ESAD represents the four high-order bits of the 36-bit source address.

### 16.3.1.5 Source Address Registers (SAR<sub>n</sub>)

The source address registers, shown in Figure 16-10, contain the address from which the DMA controller reads data. In direct mode, if MR<sub>n</sub>[CDSM/SWSM] and MR<sub>n</sub>[SRW] are set, a write to this register simultaneously sets MR<sub>n</sub>[CS], starting a DMA transfer. Software must ensure that this is a valid address.



**Figure 16-10. Source Address Registers (SAR $n$ )**

Table 16-10 describes the field of the SAR $n$ .

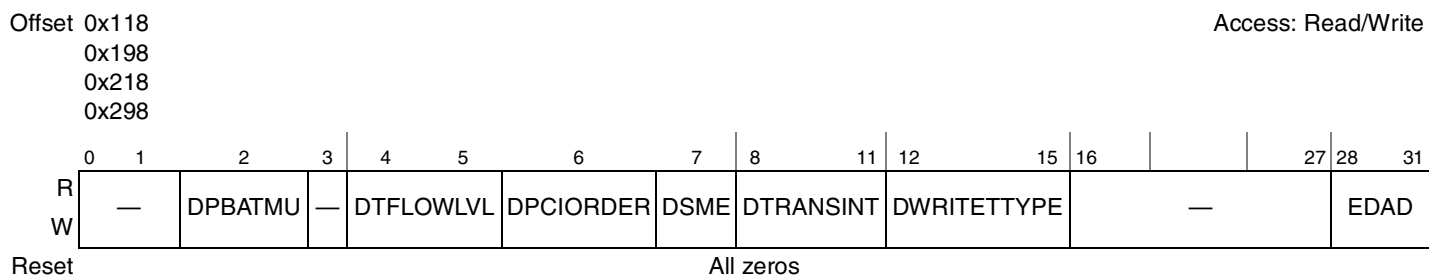
**Table 16-10. SAR $n$  Field Descriptions**

Bits	Name	Description
0–31	SAD	Source address. This register contains the low-order bits of the 36-bit source address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

### 16.3.1.6 Destination Attributes Registers (DATR $n$ )

The destination attributes registers, shown in Figure 16-11, contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATR $n$ [DSME].

If DATR $n$ [DBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DATR $n$ [DWRITETYPE] using the local address space category.



**Figure 16-11. Destination Attributes Registers (DATR $n$ )**

Table 16-11 describes the fields of the  $DATR_n$ .

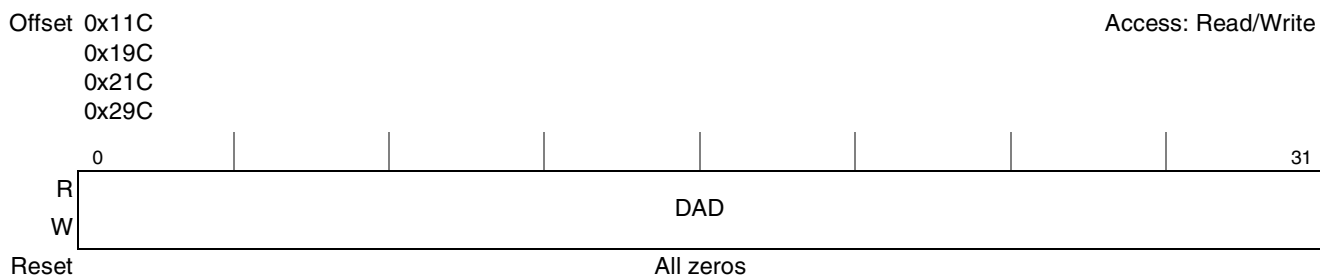
**Table 16-11.  $DATR_n$  Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2	DBPATMU	Bypass ATMU for this DMA operation 0 Route the operation through the ATMU outbound windows. $DATR_n[DWRITETYPE]$ should specify a local address space transaction type. 1 Bypass ATMU. Never generate an address match. Always use the attributes in the destination attribute registers. Route the transaction to the interface specified in the $DATR_n[DTRANSINT]$ field. Applicable only to RapidIO interface.
3	—	Reserved
4–5	DTFLOWLVL	RapidIO transaction flow level 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved Applicable only to RapidIO interface, while $DATR_n[DBPATMU]$ is set.
6	DPCI_ORDER	PCI ordering rules enable. Applicable only while $DATR_n[DBPATMU]$ is set. 0 Retain original transaction ordering 1 Follow PCI transaction ordering rules on RapidIO (elevate write priority one level over reads).
7	DSME	Destination stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode ( $MR_n[XFE]$ is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in $DSR_n$ .
8–11	—	Reserved
12–15	DWRITETYPE	DMA destination transaction type. Reserved values will result in a programming error being detected and logged in $SR[PE]$ . Transaction type to run on local address space 0000–0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, allocate L2 cache line 0111 Write, allocate and lock L2 cache line 1000–1111 Reserved
16–27	—	Reserved
28–31	EDAD	Extended destination address. EDAD represents the four high-order bits of the 36-bit destination address.

### 16.3.1.7 Destination Address Registers ( $DAR_n$ )

The destination address registers, shown in Figure 16-12, contain the addresses to which the DMA controller writes data.

In direct mode, if  $MR_n[SRW]$  is set and  $MR_n[CDSM/SWSM]$  is cleared, a write to this register simultaneously sets  $MR_n[CS]$ , starting a DMA transfer. Software must ensure that this is a valid address.



**Figure 16-12. Destination Address Registers (DAR $n$ )**

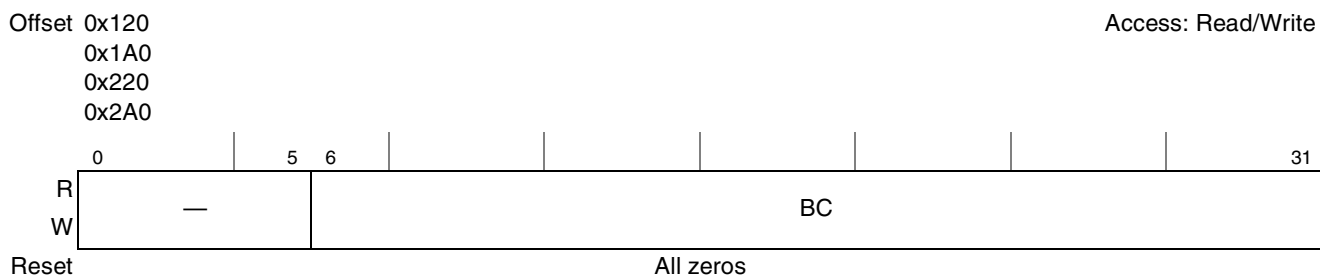
Table 16-12 describes the field of the DAR $n$ .

**Table 16-12. DAR $n$  Field Descriptions**

Bits	Name	Description
0–31	DAD	Destination address. This register contains the low-order bits of the 36-bit destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

### 16.3.1.8 Byte Count Registers (BCR $n$ )

The byte count register, shown in Figure 16-13, contains the number of bytes to transfer.



**Figure 16-13. Byte Count Registers (BCR $n$ )**

Table 16-13 describes the fields of the BCR $n$ .

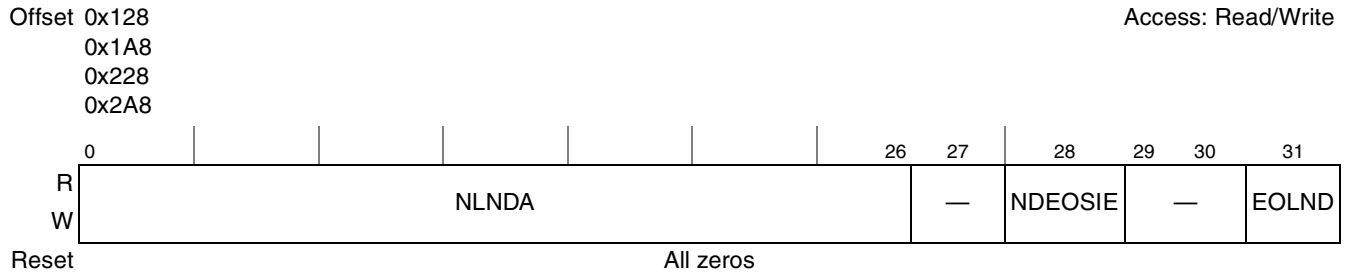
**Table 16-13. BCR $n$  Field Descriptions**

Bits	Name	Description
0–5	—	Reserved
6–31	BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is $(2^{26}) - 1$ bytes.

### 16.3.1.9 Next Link Descriptor Address Registers (NLNDAR $n$ and ENLNDAR $n$ )

The next link descriptor address registers, shown in Figure 16-14 and Figure 16-15, contain the address for the next link descriptor in memory. Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

DMA Controller

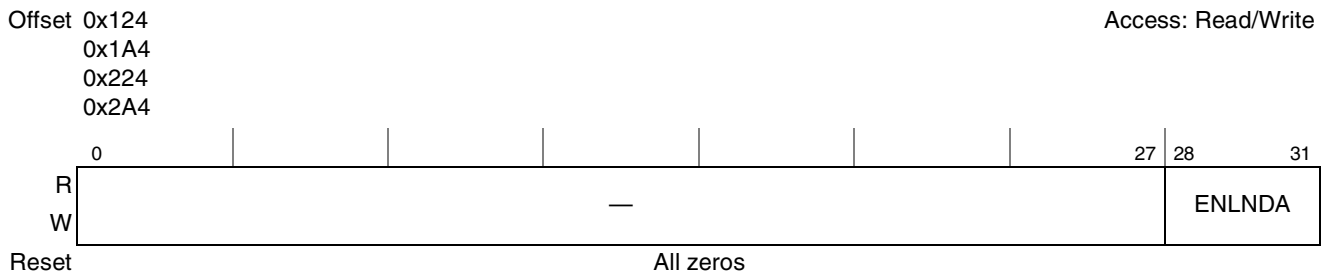


**Figure 16-14. Next Link Descriptor Address Registers (NLNDAR<sub>n</sub>)**

Table 16-14 describes the fields of the NLNDAR<sub>n</sub> registers.

**Table 16-14. NLNDAR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–26	NLNDA	Next link descriptor address. Contains the next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.
27	—	Reserved
28	NDEOSIE	Next descriptor end-of-segment interrupt enable 0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished. 1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.
29–30	—	Reserved
31	EOLND	End-of-links descriptor. This bit is ignored in direct mode. 0 This descriptor is not the last link descriptor in memory for this list. 1 This descriptor is the last link descriptor in memory for this list. If this bit is set, the DMA controller advances to the next list descriptor in memory if NLSDAR <sub>n</sub> [EOLSD] is also set in extended mode.



**Figure 16-15. Extended Next Link Descriptor Address Registers (ENLNDAR<sub>n</sub>)**

Table 16-15 describes the fields of the ENLNDAR<sub>n</sub> registers.

**Table 16-15. ENLNDAR<sub>n</sub> Field Descriptions**

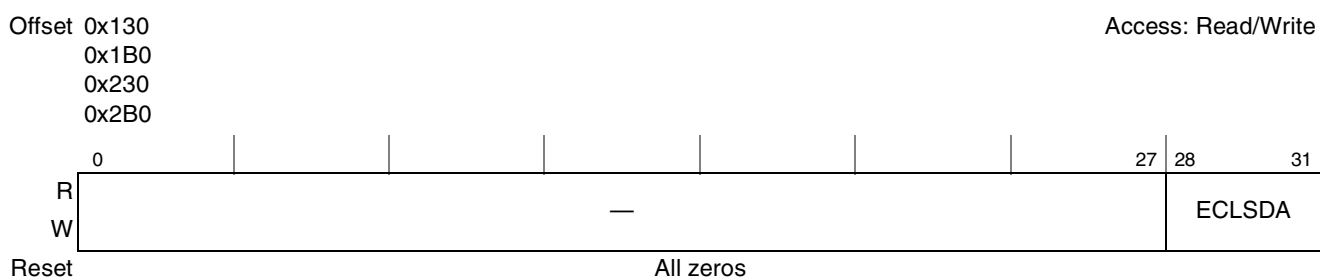
Bit	Name	Description
0–27	—	Reserved
28–31	ENLNDA	Next link descriptor extended address bits (upper 4 bits of 36-bit address)



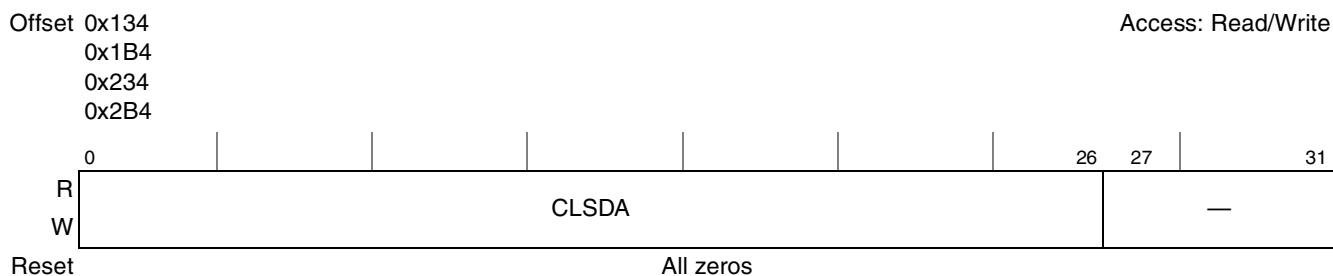
### 16.3.1.10 Current List Descriptor Address Registers (CLSDAR $n$ and ECLSDAR $n$ )

The current list descriptor address registers, shown in [Figure 16-17](#) and [Table 16-16](#), contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR $n$  and ECLSDAR $n$  to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR $n$ [EOLSD] in the next list descriptor address register is clear, the DMA controller reads the new current list descriptor from memory to process that list. If EOLSD in the next list descriptor address register is set and the last link in the current list is finished all DMA transfers are complete.



**Figure 16-16. Extended Current List Descriptor Address Registers (ECLSDAR $n$ )**



**Figure 16-17. Current List Descriptor Address Registers (CLSDAR $n$ )**

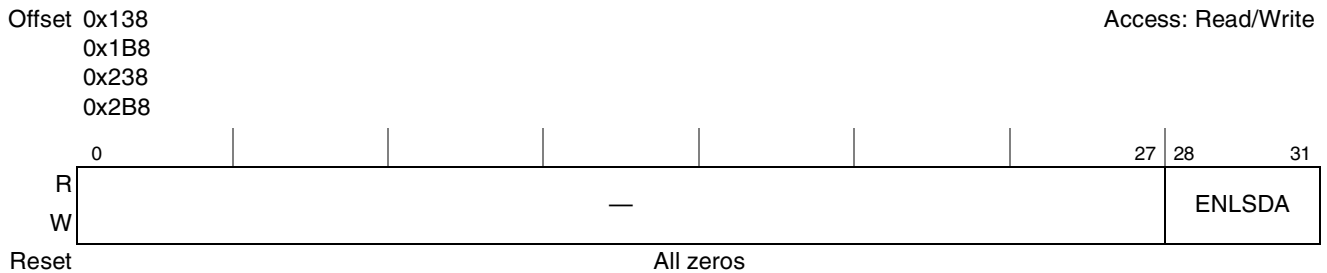
[Table 16-16](#) describes the fields of the CLSDAR $n$ .

**Table 16-16. CLSDAR $n$  Field Descriptions**

Bits	Name	Description
0–26	CLSDA	Current list descriptor address. Contains the low-order bits of the 36-bit current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31	—	Reserved

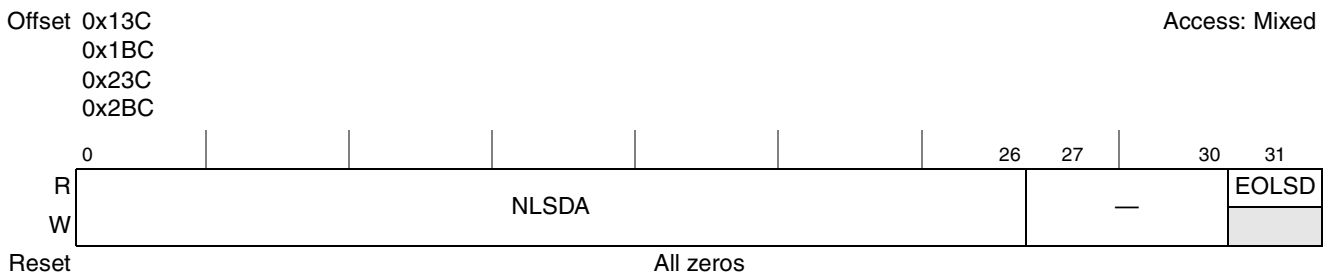
### 16.3.1.11 Next List Descriptor Address Registers (NLSDA $n$ and ENLSDA $n$ )

The next list descriptor address registers, shown in [Figure 16-18](#) and [Figure 16-19](#), contain the address for the next list descriptor in memory. If the contents are transferred to the current list descriptor address register they become effective for the current transfer in extended chaining mode.



**Figure 16-18. Extended Next List Descriptor Address Registers (ENLSDA $n$ )**

[Table 16-17](#) describes the fields of the NLSDA $n$ .



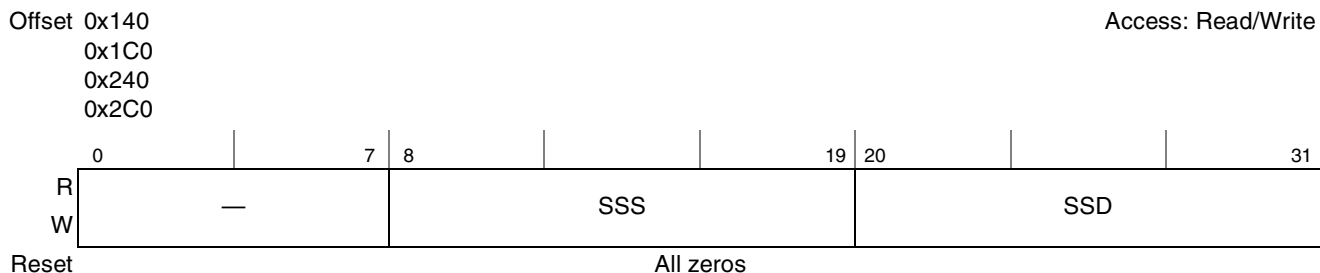
**Figure 16-19. Next List Descriptor Address Registers (NLSDA $n$ )**

**Table 16-17. NLSDA $n$  Field Descriptions**

Bits	Name	Description
0–26	NLSDA	Next list descriptor address. Contains the low-order bits of the 36-bit next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30	—	Reserved
31	EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

### 16.3.1.12 Source Stride Registers (SSR $n$ )

The source stride register, shown in [Figure 16-20](#), contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.



**Figure 16-20. Source Stride Registers (SSR $n$ )**

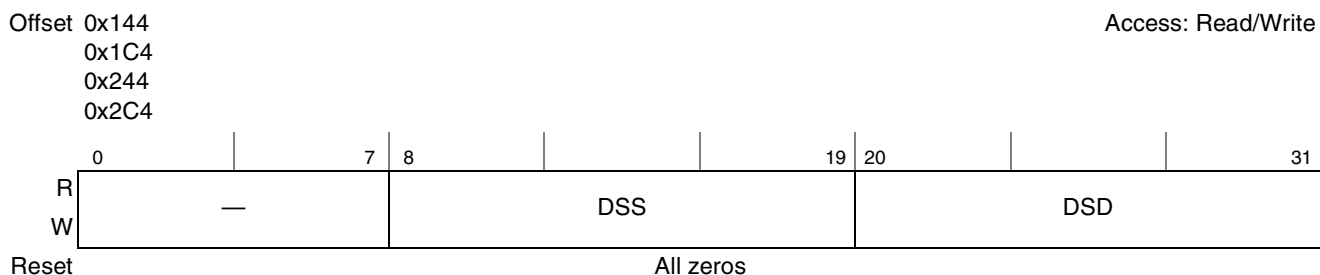
Table 16-18 describes the fields of the SSR $n$ .

**Table 16-18. SSR $n$  Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–19	SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31	SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

### 16.3.1.13 Destination Stride Registers (DSR $n$ )

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated. Figure 16-21 describes the DSR $n$ .



**Figure 16-21. Destination Stride Registers (DSR $n$ )**

Table 16-19 describes the fields of the DSR $n$ .

**Table 16-19. DSR $n$  Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–19	DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31	DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

### 16.3.1.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register. This register is read-only. Figure 16-22 describes the DGSR.

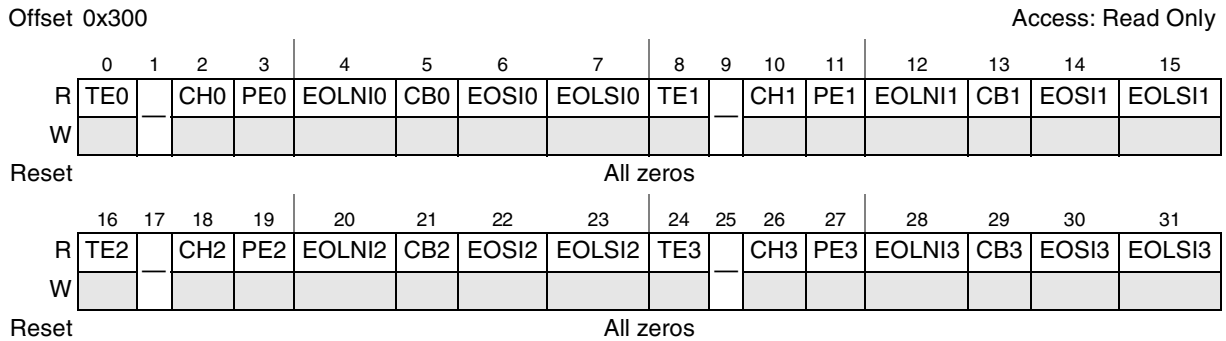


Figure 16-22. DMA General Status Register (DGSR)

Table 16-20 describes the fields of the DGSR.

Table 16-20. DGSR Field Descriptions

Bits	Name	Description
0	TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1	—	Reserved
2	CH0	Channel halted, channel 0
3	PE0	Programming error, channel 0
4	EOLNI0	End-of-links interrupt, channel 0
5	CB0	Channel busy, channel 0
6	EOSI0	End-of-segment interrupt, channel 0
7	EOLSI0	End-of-lists/direct interrupt, channel 0
8	TE1	Transfer error, channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9	—	Reserved
10	CH1	Channel halted, channel 1
11	PE1	Programming error, channel 1
12	EOLNI1	End-of-links interrupt, channel 1
13	CB1	Channel busy, channel 1
14	EOSI1	End-of-segment interrupt, channel 1
15	EOLSI1	End-of-lists/direct interrupt, channel 1

Table 16-20. DGSR Field Descriptions (continued)

Bits	Name	Description
16	TE2	Transfer error, channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17	—	Reserved
18	CH2	Channel halted, channel 2
19	PE2	Programming error, channel 2
20	EOLNI2	End-of-links interrupt, channel 2
21	CB2	Channel busy, channel 2
22	EOSI2	End-of-segment interrupt, channel 2
23	EOLSI2	End-of-lists/direct interrupt, channel 2
24	TE3	Transfer error, channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25	—	Reserved
26	CH3	Channel halted, channel 3
27	PE3	Programming error, channel 3
28	EOLNI3	End-of-links interrupt, channel 3
29	CB3	Channel busy, channel 3
30	EOSI3	End-of-segment interrupt, channel 3
31	EOLSI3	End-of-lists/direct interrupt, channel 3

## 16.4 Functional Description

This section describes the function of the DMA controller.

### 16.4.1 DMA Channel Operation

All DMA channels support two different modes of operation: a basic mode ( $MR_n[XFE]$  is cleared) and an extended mode ( $MR_n[XFE]$  is set). In both modes, a channel can be activated by clearing and setting  $MR_n[CS]$ , or through the single-write start mode using  $MR_n[CDSM/SWSM]$  and  $MR_n[SRW]$ , or through an external control mode using  $MR_n[ECS\_EN]$ .

In basic mode, the channel can be programmed in basic direct mode or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode or extended chaining mode. Extended mode provides more capabilities, such as extended descriptor chaining, striding capabilities, and a more flexible descriptor structure.

The DMA controller supports misaligned transfers for both the source and destination addresses. In order to maximize performance, the source and destination engines align the source and destination addresses to a 64-byte boundary. The DMA always reads/writes the maximum number of bytes for a given transfer as

described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input.

The DMA controller supports bandwidth control, which prevents a channel from consuming all the data bandwidth in the controller. Each channel is allowed to consume the bandwidth of the shared resources as specified by the bandwidth control value. After the channel uses its allotted bandwidth, the arbiter grants the next channel access to the shared resources. The arbitration is round robin between the channels. This feature is also used to implement the external control pause feature. If the external control start and pause are enabled in the  $MR_n$ , the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit,  $MR_n[CC]$ . See [Table 16-21](#) for more complete descriptions of the channel states and state transitions.

### 16.4.1.1 Basic DMA Mode Transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset. The different modes of operation under the basic mode are explained in the following sections.

#### 16.4.1.1.1 Basic Direct Mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing  $SAR_n$ ,  $SATR_n$ ,  $DAR_n$ ,  $DATR_n$ , and  $BCR_n$  registers. The DMA transfer is started when  $MR_n[CS]$  is set. Software is expected to program all the appropriate registers before setting  $MR_n[CS]$  to a 1. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see [Table 16-21](#)), to confirm that the specific DMA channel is idle.
2. Initialize  $SAR_n$ ,  $SATR_n$ ,  $DAR_n$ ,  $DATR_n$  and  $BCR_n$ .
3. Set the mode register channel transfer mode bit,  $MR_n[CTM]$ , to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear then set the mode register channel start bit,  $MR_n[CS]$ , to start the DMA transfer.
5.  $SR_n[CB]$  is set by the DMA controller to indicate the DMA transfer is in progress.
6.  $SR_n[CB]$  is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted ( $MR_n[CA]$  transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if  $MR_n[EOSIE]$  is set.

#### 16.4.1.1.2 Basic Direct Single-Write Start Mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing the  $SATR_n$ ,  $DATR_n$ , and  $BCR_n$  registers. Setting  $MR_n[SRW]$  configures the

DMA controller to begin the DMA transfer either when  $SAR_n$  is written or when  $DAR_n$  is written, determined by the state of  $MR_n[CDSM/SWSM]$ . Writing to  $SAR_n$  initiates the DMA transfer if  $MR_n[CDSM/SWSM]$  is set. Writing to  $DAR_n$  initiates the DMA transfer if  $MR_n[CDSM/SWSM]$  is cleared. The DMA controller automatically sets the channel start bit,  $MR_n[CS]$ . Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see [Table 16-21](#)), to confirm that the specific DMA channel is idle.
2. Initialize the source attributes ( $SATR_n$ ),  $DATR_n$ , and  $BCR_n$  registers.
3. Set the mode register channel transfer mode bit,  $MR_n[CTM]$ , and the single-write start direct mode bit,  $MR_n[SRW]$ . Other control parameters may also be initialized in the mode register. Set  $MR_n[CDSM/SWSM]$  for transfers started using  $SAR_n$ . Clear  $MR_n[CDSM/SWSM]$  for transfers started using the  $DAR_n$ .
4. A write to the source or destination address register starts the DMA transfer and automatically sets  $MR_n[CS]$ .
5.  $SR_n[CB]$  is set by the DMA controller to indicate the DMA transfer is in progress.
6.  $SR_n[CB]$  is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted ( $MR_n[CA]$  transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if  $MR_n[EOSIE]$  is set.

#### 16.4.1.1.3 Basic Chaining Mode

In basic chaining mode, software must first build link descriptor segments in memory. Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. The transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory.
2. Poll the channel state (see [Table 16-21](#)), to confirm that the specific DMA channel is idle.
3. Initialize  $CLNDAR_n$  and  $ECLNDAR_n$  to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit,  $MR_n[CTM]$ , as well as  $MR_n[XFE]$ , to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit,  $MR_n[CS]$ , to start the DMA transfer.
6.  $SR_n[CB]$  is set by the DMA controller to indicate the DMA transfer is in progress.
7.  $SR_n[CB]$  is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted ( $MR_n[CA]$  transitions from a 0 to 1), or if an error occurs during any of the transfers.

#### 16.4.1.1.4 Basic Chaining Single-Write Start Mode

Basic chaining single-write start mode allows a chain to be started by writing the current link descriptor address register (CLNDAR $n$ ). (Note that ECLNDAR $n$  must be written *first* so that the full 36-bit descriptor address is present when the chain starts.) Setting MR $n$ [CDSM/SWSM] in the mode register causes MR $n$ [CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR $n$ [CDSM/SWSM], and the extended features enable bit MR $n$ [XFE]. Also, clear the channel transfer mode bit, MR $n$ [CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see Table 16-21), to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR $n$  and ECLNDAR $n$  to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR $n$ [CS].
5. SR $n$ [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR $n$ [CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR $n$ [CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

#### 16.4.1.2 Extended DMA Mode Transfer

The extended DMA mode also operates in chaining and direct mode. It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR $n$ [XFE].

##### 16.4.1.2.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR $n$ [XFE] being set. Striding on the source address can be accomplished by setting SATR $n$ [SSME] and setting the desired stride size and distance in SSR $n$ . Striding on the destination address can be accomplished by setting DATR $n$ [DSME] and setting the desired stride size and distance in DSR $n$ .

##### 16.4.1.2.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR $n$ [XFE] being set. Striding on the source address can be accomplished by setting SATR $n$ [SSME] and setting the desired stride size and distance in SSR $n$ . Striding on the destination address can be accomplished by setting DATR $n$ [DSME] and setting the desired stride size and distance in DSR $n$ .



### 16.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR<sub>n</sub> and ECLSDAR<sub>n</sub> must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see Table 16-21), to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR<sub>n</sub> and ECLSDAR<sub>n</sub> to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR<sub>n</sub>[CTM], to indicate chaining mode. MR<sub>n</sub>[XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR<sub>n</sub>[CS], to start the DMA transfer.
6. SR<sub>n</sub>[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR<sub>n</sub>[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR<sub>n</sub>[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

### 16.4.1.2.4 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR<sub>n</sub>[CDSM/SWSM] causes MR<sub>n</sub>[CS] to be set automatically when CLSDAR<sub>n</sub> is written. (Note that ECLSDAR<sub>n</sub> must be written *first* so that the full 36-bit descriptor address is present when the chain starts.) The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR<sub>n</sub>[CDSM/SWSM], MR<sub>n</sub>[CTM], and MR<sub>n</sub>[XFE] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see Table 16-21), to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR<sub>n</sub>[CS].
5. SR<sub>n</sub>[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR<sub>n</sub>[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR<sub>n</sub>[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

### 16.4.1.3 External Control Mode Transfer

An external control can be used to control all DMA channels by setting  $MR_n[EMS\_EN]$ . The external control can control the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

Note that when operating the DMA in chaining mode the register byte count field,  $BCR[BC]$ , must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or restart a paused DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion.

The pause feature can be enabled by setting  $MR_n[EMP\_EN]$ .  $MR_n[BWC]$  specifies how much data to allow a specific channel to transfer before entering a paused state by clearing  $MR_n[CS]$ . Note, however, that write data for a paused transfer may not have reached the target interface when so indicated. The channel can be restarted from a paused state by the asserted edge of  $\overline{DMA\_DREQ}$  as driven by an external master. In chaining modes, the channel does not pause for descriptor fetch transfer. It only pauses during the actual data transfer.

The following signals are defined for the external control interface:

- $\overline{DMA\_DREQ}$  - Asserting edge triggers a DMA transfer start or restart from a pause request. Sets  $MR_n[CS]$ .
- $\overline{DMA\_DACK}$  - Indicates a DMA transfer currently in progress.  $SR_n[CB]$  is set.
- $\overline{DMA\_DDONE}$  - Indicates the completion of the DMA controller's involvement in the transfer and the readiness to accept a new DMA command.  $SR_n[CB]$  is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in [Table 16-3](#). The timing diagram of the external control interface is shown in [Figure 16-23](#).

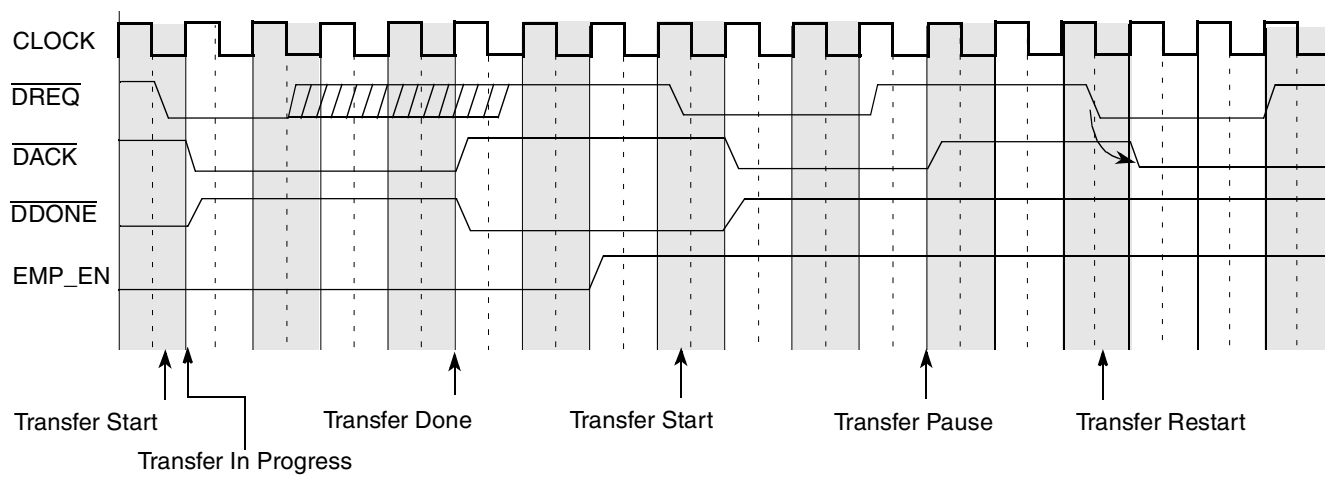


Figure 16-23. External Control Interface Timing

#### 16.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode (enabled when  $MR_n[CC]$  is set) offers software the flexibility of having the DMA controller get started on descriptors that have already been programmed while software continues to build more descriptors in memory. Software can set the end-of-links descriptor (EOLND) in basic mode, or end-of-lists descriptor (EOLSD) in extended mode, to cause the channel to go into a halted state while software continues to build other descriptors in memory. Software can then set  $CC$  to force hardware to continue where it left off. Channel continue is only meaningful for chaining modes, not direct mode.

If  $CC$  is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor.

If  $CC$  is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

##### 16.4.1.4.1 Basic Mode

On a channel continue, the descriptor at the current link descriptor address registers ( $CLNDAR_n$  and  $ECLNDAR_n$ ) is refetched to get the next link descriptor address field as updated by software. The channel halts if  $NLNDAR_n[EOLND]$  is still set. If EOLND is zero, the next link descriptor address is copied into  $CLNDAR_n$  and  $ECLNDAR_n$  and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

### 16.4.1.4.2 Extended Mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR $n$  and ECLSDAR $n$ ) address register is refetched to get the next list descriptor address field as updated by software. The channel halts if NLSDAR $n$ [EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR $n$  and ECLSDAR $n$  registers and the channel continues with another descriptor fetch of the current list descriptor address. As a result, two list descriptor fetches always exist after channel continue before the first link descriptor fetch and the first transfer.

### 16.4.1.5 Channel Abort

Software can abort a previously initiated transfer by setting MR $n$ [CA]. Once the DMA channel controller detects a zero-to-one transition of MR $n$ [CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR $n$ [CB]. Successful completion of a software initiated abort request can be recognized by MR $n$ [CA] being set and SR $n$ [CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR $n$ [TE] is set), or the channel has completed all transfers, then SR $n$ [CB] being cleared may not signify that the controller entered a halt state due to the abort request.

### 16.4.1.6 Bandwidth Control

MR $n$ [BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware. This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 Kbyte at a time when no other channel is active.

### 16.4.1.7 Channel State

Table 16-21 defines the state of a channel based on the values of the channel start (MR $n$ [CS]), channel busy (SR $n$ [CB]), transfer error (SR $n$ [TE]), and channel continue (MR $n$ [CC]) bits.

Table 16-21. Channel State Table

MR $n$ [CS]	SR $n$ [CB]	SR $n$ [TE]	MR $n$ [CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared MR $n$ [CS].
0	1	0	1	Channel remains in halt state.
—	1	1	—	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.

Table 16-21. Channel State Table (continued)

MRn[CS]	SRn[CB]	SRn[TE]	MRn[CC]	Channel State
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state
1	1	0	0	Transfer in progress
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue

### 16.4.1.8 Illustration of Stride Size and Stride Distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance. The stride distance is added to the current base address to point to the next quantity of data to be transferred. Figure 16-24 illustrates the stride size and distance parameters. As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

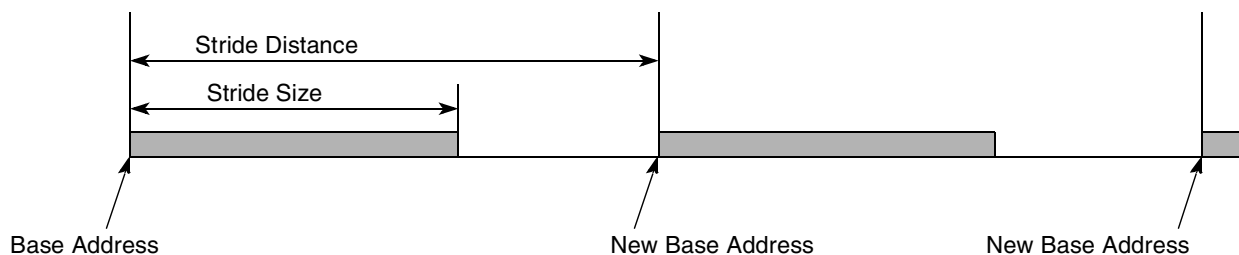


Figure 16-24. Stride Size and Stride Distance

## 16.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

## 16.4.3 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, address mapping errors, for example), the DMA halts by setting SRn[TE] and generates an interrupt if MRn[EIE] is set. On a programming error, the DMA sets SRn[PE] and generates an interrupt if MRn[EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero
- Stride transfer started with a stride size of zero
- Transfer started with a priority of three
- Illegal type, defined by SATRn[SREADTTYPE] and DATRn[DWRITETTYPE], used for the transfer.

## 16.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link descriptors. List descriptors connect lists of link descriptors. Link descriptors describe the DMA activity that is to take place. DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the NLNDAR<sub>n</sub>[EOLND] bit in the next link descriptor; and NLSDAR<sub>n</sub>[EOLSD] in the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

### NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

Table 16-22 summarizes the DMA list descriptors.

**Table 16-22. List DMA Descriptor Summary**

Descriptor Field	Description
Next list descriptor extended address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor extended address registers.
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor extended address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor extended address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

Table 16-23 summarizes the DMA link descriptors.

**Table 16-23. Link DMA Descriptor Summary**

Descriptor Field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor extended address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the extended next link descriptor address registers
Next link descriptor address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the next link descriptor address registers.
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the byte count register.

Figure 16-25 describes the DMA transaction flow.

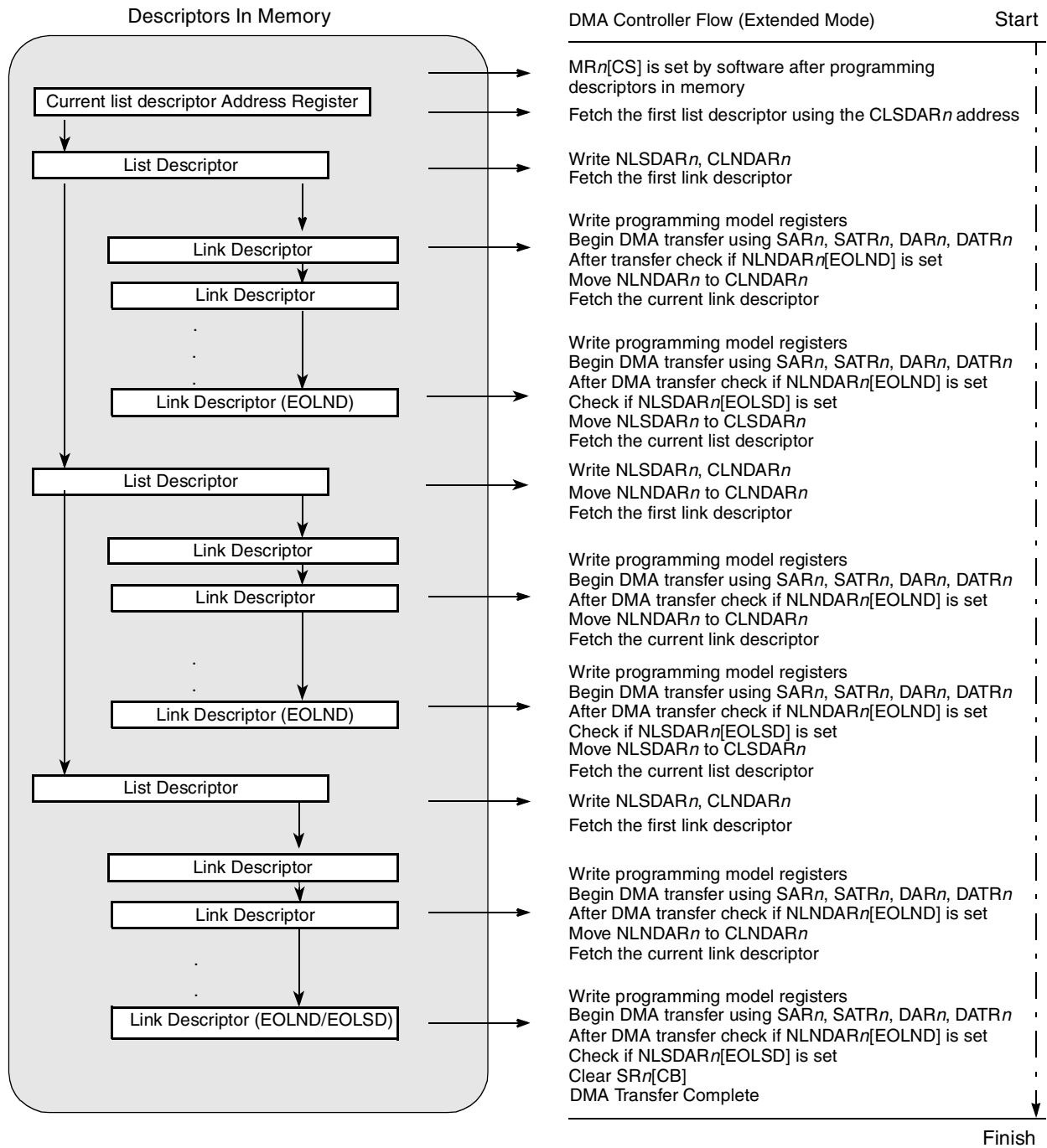


Figure 16-25. DMA Transaction Flow with DMA Descriptors



Figure 16-26 describes the format of the list descriptors.

**Figure 16-26. List Descriptor Format**

Offset	
0x00	Next List Descriptor Extended Address
0x04	Next List Descriptor Address
0x08	First Link Descriptor Extended Address
0x0C	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1C	Reserved

Figure 16-27 describes the format of the link descriptors.

**Figure 16-27. Link Descriptor Format**

Offset	
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0C	Destination Address
0x10	Next Link Descriptor Extended Address
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1C	Reserved

## 16.4.5 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, stride sizes less than 64 bytes should be avoided. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are as follows:

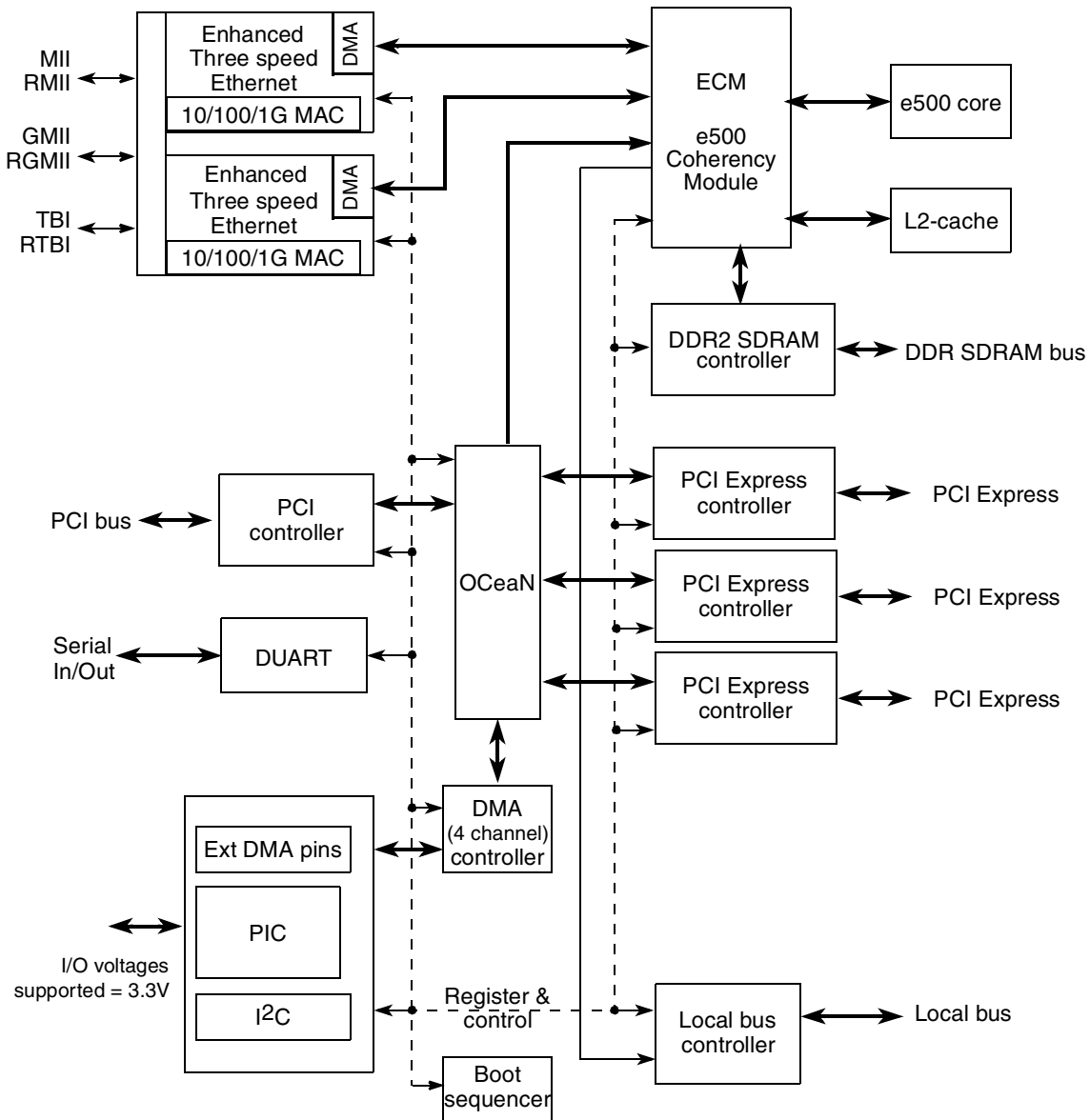
- All interface capabilities from where descriptors are being fetched must support read sizes of 32 bytes or greater.
- If  $MRn[SAHE]$  is set, the source interface transfer size capability must be greater than or equal to  $MRn[SAHTS]$ . The source address must be aligned to a size specified by SAHTS.

- If  $MRn[DAHE]$  is set, the destination interface transfer size capability must be greater than or equal to  $MRn[DAHTS]$ . The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if  $MRn[DAHE]$  is set and source striding is not supported if  $MRn[SAHE]$  is set.

## 16.5 DMA System Considerations

Figure 16-28 shows the most important data paths within the device. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller. This section provides information about how to make most effective use of these DMA channels including the following topics:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources and destinations
- Transparency of the bus interfaces to DMA operations
- What is useful as opposed to what is possible. For example, it is possible to address any internal register through the internal control bus, which means configuration and control registers can be DMA source or destination targets. However, the typical use of DMA functionality is to reduce host processor loading by moving large amounts of data with minimal CPU involvement. Using a general-purpose DMA controller to load small amounts of configuration data only makes sense in special circumstances (perhaps during system boot, for example).



**Figure 16-28. DMA Data Paths**

**Note:** On-chip target configuration registers include I<sup>2</sup>C data register.

**Note:** On-chip Ethernet captive resource. Not available to external masters.

**Note:** On-chip 4-channel controller can serve external masters.

## 16.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks cannot serve as DMA targets. The following topics are addressed:

- Transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

### 16.5.1.1 DMA to e500 Core

The L1 cache cannot be a direct DMA target because it cannot be directly addressed by software. However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

### 16.5.1.2 DMA to Ethernet

The Ethernet controllers cannot serve as DMA targets because they have no suitable internal memory for this purpose. The Ethernet controllers have dedicated DMA channels to move data between the external transmit and receive buffers and the internal packet buffer. This dedicated channel is the only DMA service to the internal packet buffers.

However, Ethernet ports can serve as DMA targets by using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

### 16.5.1.3 DMA to Configuration, Control, and Status Registers

Because any internal register can be addressed with the four-channel DMA controller, configuration, control, and status registers throughout the device are valid DMA targets. However, the primary purpose of DMA—to reduce processor load by moving large blocks of data—is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I<sup>2</sup>C controller or programmable interrupt controller (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

### 16.5.1.4 DMA to I<sup>2</sup>C

The I<sup>2</sup>C controller is not transparent to DMA transfers. Observe the caveats listed in [Section 16.5.1.3, “DMA to Configuration, Control, and Status Registers,”](#) when accessing any I<sup>2</sup>C register, including the data register (I2CDR).

### 16.5.1.5 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in [Chapter 13](#), “DUART,” specifically, [Section 13.4.5](#), “FIFO Mode.”



## Chapter 17

# PCI Bus Interface

The PCI interface complies with the *PCI Local Bus Specification*, Rev. 2.2. It is beyond the scope of this manual to document the intricacies of the PCI bus. This chapter describes the PCI controller (referenced as PCI throughout this chapter) of this device and provides a basic description of PCI bus operations. The specific emphasis is directed at how the integrated processor implements the PCI bus. Designers of systems incorporating PCI devices should refer to the specification for a thorough description of the PCI bus.

### NOTE

Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Because this is inconsistent with the terminology in this manual, the terms ‘word’ and ‘double word’ are not used in this chapter. Instead, the number of bits or bytes indicates the exact quantity.

## 17.1 Introduction

The PCI controller acts as a bridge between the PCI interface and the OCeaN switch fabric. Figure 17-1 is a high-level block diagram of the PCI controller.

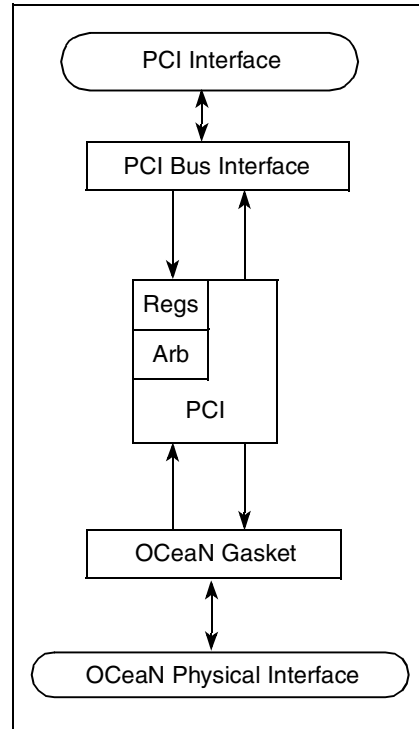


Figure 17-1. PCI Controller Block Diagram

### 17.1.1 Overview

The PCI controller connects the OCeaN to the PCI bus, to which I/O components are connected. The PCI bus uses a 32-bit multiplexed address/data bus, plus various control and error signals. The PCI interface supports address and data parity with error checking and reporting.

The integrated processor's PCI interface functions both as a master (initiator) and a target device. Internally, the design is divided into the following:

- Data path blocks
- Control logic blocks
- Memory

The data path blocks contain the queues, tables for transaction tracking, and ordering. The control blocks contain control logic and state-machines for buffer control, bus protocol, tag generation, and transaction resizing. The memory blocks are used solely for inbound and outbound data storage. This allows the integrated processor to handle separate PCI transactions simultaneously. For example, consider the case where a burst-write transaction from the integrated processor to another PCI device terminates with a



disconnect before finishing the transaction. If another PCI device is granted the PCI bus and requests a burst-read from local memory, the integrated processor, as a target, can accept the burst-read transfer. When the integrated processor is granted mastership of the PCI bus, the burst-write transaction continues. The PCI interface does not flush pending outbound writes as a result of an inbound read command. Systems must not rely on inbound reads to ensure all pending outbound writes have completed. For example, consider the case where a core writes data to a PCI device and then updates a flag in the local DDR memory indicating the write to PCI has completed. An external PCI master may misread the flag ahead of the actual write transaction's completion on the PCI bus.

There are two blocks of memory in the design:

- The inbound buffers
- The outbound read buffers combined with the outbound write buffers

There are many blocks of control logic in the block. On the PCI side there are machines for PCI controller initiated address and data tenures for inbound and outbound data, respectively. On the OCeaN side there are machines for fabric arbitration, outbound data, and inbound data.

As an initiator, the integrated processor supports read and write operations to the PCI memory space, the PCI I/O space, and the 256-byte PCI configuration space. As an initiator, the integrated processor also supports generating PCI special-cycle and interrupt-acknowledge transactions. As a target, the integrated processor supports read and write operations to local memory, and, when configured in agent mode, read and write operations to the internal PCI configuration registers.

The integrated processor can function as either a PCI host bridge (host mode) or a peripheral device on the PCI bus (agent mode). See [Section 17.1.3.1.1, “Host Mode,”](#) for more information.

In agent mode, all of the PCI configuration registers in the integrated processor can be programmed from the PCI bus. See [Section 17.4.2.11.3, “Agent Accessing the PCI Configuration Space,”](#) for more information.

The PCI interface provides bus arbitration for the integrated processor and up to five other PCI bus masters. The arbitration algorithm is a programmable two-level, round-robin priority selector. The on-chip PCI arbiter can operate in both host and agent modes or it can be disabled to allow for an external PCI arbiter.

The integrated processor also provides an address translation mechanism to map inbound PCI to OCeaN accesses and outbound OCeaN to PCI accesses.

### 17.1.1.1 Outbound Transactions

Upon detecting an OCeaN-to-PCI transaction, the integrated processor requests the use of the PCI bus. For OCeaN-to-PCI bus write operations, the integrated processor requests mastership of the PCI bus when the source completes the write operation to the OCeaN. For OCeaN-to-PCI read operations, the integrated processor requests mastership of the PCI bus when it decodes that the access is for PCI address space.

Once granted, the integrated processor drives the address (PCI\_AD[31:0]) and the bus command (PCI\_C/ $\overline{\text{BE}}$ [3:0]) signals.

The master part of the interface can initiate master-abort cycles, recognizes target-abort, target-retry, and target-disconnect cycles, and supports various device selection timings. The master interface does not run fast back-to-back or exclusive accesses.

### 17.1.1.2 Inbound Transactions

Upon detection of a PCI address phase, the integrated processor decodes the address and bus command to determine if the transaction is within the local memory access boundaries. If the transaction is destined for local memory, the target interface latches the address, decodes the PCI bus command, and forwards the transaction to the OCeaN control unit. On writes to local memory, data is forwarded along with the byte enables (if applicable) to the internal control unit. Note that for inbound writes less than 4-bytes, the PCI controller splits the transaction into single byte writes to the target. Thus, the PCI interface cannot be used to perform single beat writes to 16-bit devices on the local bus interface. On reads, the data is driven on the bus and the byte enables (if applicable) determine which byte lanes contain meaningful data.

The target interface of the integrated processor can issue target-abort, target-retry, and target-disconnect cycles. The target interface supports fast back-to-back transactions. The target interface uses the fastest device selection timing.

The integrated processor supports data streaming to and from local memory. This means that data can flow between the processor PCI interface and local memory as long as the internal buffers are not filled.

## 17.1.2 Features

The following is a list of PCI features that is supported:

- PCI interface 2.2 compatible
- 66- and 33-MHz support
- 32-bit PCI interface support on PCI port
- Host and agent mode support
- 64-bit dual address cycle (DAC) support
- On-chip arbitration with support for five high-priority request and grant signal pairs
- Support for accesses to all PCI memory and I/O address spaces
- Support for PCI-to-memory and memory-to-PCI streaming
- Memory prefetching of PCI read accesses
- Support posting of processor-to-PCI and PCI-to-memory writes
- Support selectable snoop for inbound accesses
- PCI configuration registers
- PCI 3.3-V compatible

## 17.1.3 Modes of Operation

A number of parameters that affect the PCI controller modes of operation are determined at power-on reset (POR) by reset configuration signals as described in [Chapter 4, “Reset, Clocking, and Initialization.”](#) [Table 17-1](#) provides a summary of these modes.

Table 17-1. POR Parameters for PCI Controller

Parameter	Description	Section/page
Host/agent configuration	Selects between host and agent mode for the PCI interface.	4.4.3.5/4-14
PCI clocking	Selects between asynchronous or synchronous clocking for PCI	4.4.3.14/4-19
PCI arbiter enable	Enables the on-chip PCI bus arbiter	4.4.3.17/4-20
PCI I/O impedance	Selects the impedance of the PCI I/O drivers	4.4.3.16/4-20

### 17.1.3.1 Host/Agent Mode Configuration

The PCI controller can function as either a PCI host bridge (referred to as host mode) or a peripheral device on the PCI bus (referred to as agent mode). Additionally, the PCI controller can operate in agent configuration lock mode. Note that host/agent mode selection is determined at power-up.

#### 17.1.3.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). See [Section 17.5.1.1, “Host Mode,”](#) for more information.

#### 17.1.3.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. See [Section 17.5.1.2, “Agent Mode,”](#) for more information. Note that in PCI agent mode, the PCI controller ignores all PCI memory accesses except those to the memory-mapped registers) until inbound address translation is enabled.

#### 17.1.3.1.3 Agent Configuration Lock Mode

When the device powers up in agent configuration lock mode, it retries inbound configuration accesses until the ACL bit in the PCI bus function register is cleared. See [Section 17.5.1.3, “Agent Configuration Lock Mode,”](#) for more information.

### 17.1.3.2 PCI Clocking Configuration

The interface can be configured to be clocked asynchronously with a PCI\_CLK input or synchronously with the SYSCLK input. The initial value for clocking is determined by a power-on reset configuration signal.

### 17.1.3.3 PCI Arbiter (Internal/External Arbiter) Configuration

The interface can be configured to use an on-chip or off-chip PCI arbiter. The initial value for the arbiter is determined by a power-on reset configuration signal.

### 17.1.3.4 PCI Impedance Configuration

The device has a programmable impedance for PCI bus signals. The initial value for impedance is determined by a power-on reset configuration signal.

## 17.2 External Signal Descriptions

Figure 17-2 shows the external PCI signals.

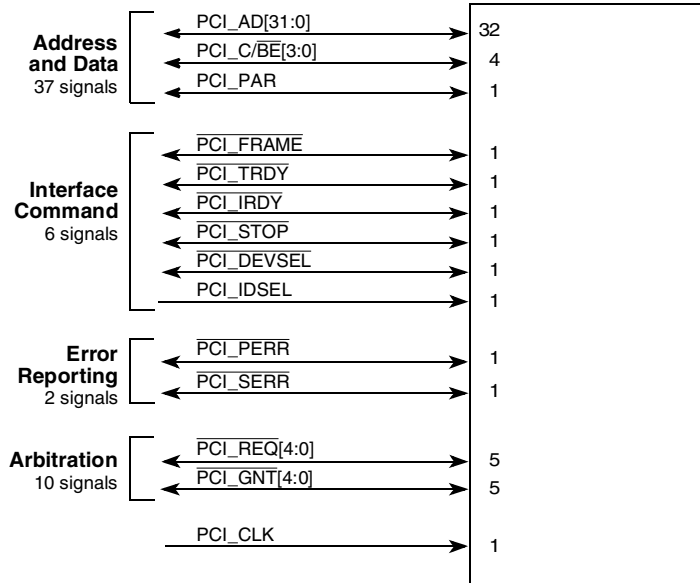


Figure 17-2. PCI Interface External Signals

Table 17-2 contains the detailed descriptions of the external PCI interface signals.

Table 17-2. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description
PCI_AD[31:0]	I/O	PCI address/data bus. The PCI address/data bus consists of signals that are both input and output signals on this PCI controller.
	O	As outputs for the bidirectional PCI address/data bus, these signals operate as described below.
	<b>State Meaning</b>	Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. The PCI_AD[7:0] signals define the LSB and PCI_AD[31:24] the MSB.
	<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional PCI address/data bus, these signals operate as described below.
	<b>State Meaning</b>	Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction. The PCI_AD[7:0] signals define the LSB and PCI_AD[31:24] the MSB.
	<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2

Table 17-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_C/ $\overline{\text{BE}}$ [3:0]	I/O	Command/byte enable. The command/byte enable signals are both input and output signals on this PCI controller. The command encodings for PCI bus mode are described in <a href="#">Section 17.4.2.2, “PCI Bus Commands.”</a>	
	O	As outputs for the bidirectional command/byte enable, these signals operate as described below.	
		<b>State Meaning</b>	Asserted/Negated—During the address phase, PCI_C/ $\overline{\text{BE}}$ [3:0] define the bus command. During the data phase, PCI_C/ $\overline{\text{BE}}$ [3:0] act as byte enables indicating which byte lanes carry meaningful data. The PCI_C/ $\overline{\text{BE}}$ [0] signal applies to the LSB.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional command/byte enable, these signals operate as described below.	
		<b>State Meaning</b>	Asserted/Negated—During the address phase, PCI_C/ $\overline{\text{BE}}$ [3:0] indicate the command that another master is sending. During the data phase, PCI_C/ $\overline{\text{BE}}$ [3:0] indicate which byte lanes are valid.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
PCI_DEVSEL	I/O	Device select. The device select signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional device select, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller has decoded the address and is the target of the current access. Negated—Indicates that this PCI controller has decoded the address and is not the target of the current access.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional device select, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that some PCI agent (other than this PCI controller) has decoded its address as the target of the current access. Negated—Indicates that no PCI agent has been selected.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
PCI_FRAME	I/O	Frame. The frame signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional frame, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller, acting as a PCI master, is initiating a bus transaction. While PCI_FRAME is asserted, data transfers may continue. Negated—If $\overline{\text{PCI\_IRDY}}$ is asserted, indicates that the PCI transaction is in the final data phase; if $\overline{\text{PCI\_IRDY}}$ is negated, indicates that the PCI bus is idle.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional frame, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that another PCI master is initiating a bus transaction. Negated—Indicates that the transaction is in the final data phase or that the bus is idle.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	

**Table 17-2. PCI Interface Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description	
$\overline{\text{PCI\_GNT}}[4:0]$	O	PCI bus grant. Output signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled $\overline{\text{PCI\_GNT}}_0$ is an input signal. Note that $\overline{\text{PCI\_GNT}}[n]$ is a point-to-point signal. Every master has its own bus grant signal. Also, note that these signals are also used as reset configuration signals as described in <a href="#">Section 4.4.3, "Power-On Reset Configuration."</a>	
		<b>State Meaning</b>	
		<b>Timing</b>	
PCI_IDSEL	I	Initialization device select. The initialization device select signal is an input signal on this PCI controller. It is used as a chip select during configuration read and write transactions.	
		<b>State Meaning</b>	
		<b>Timing</b>	
PCI_IRDY	I/O	Initiator ready. The initiator ready signal is both an input and output signal on this PCI controller.	
		O	As outputs for the bidirectional initiator ready, these signals operate as described below.
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, this PCI controller asserts $\overline{\text{PCI\_IRDY}}$ to indicate that valid data is present on the data bus. During a read, this PCI controller asserts $\overline{\text{PCI\_IRDY}}$ to indicate that it is prepared to accept data. Negated—Indicates that the PCI target needs to wait before this PCI controller, acting as a PCI master, can complete the current data phase. During a write, this PCI controller negates $\overline{\text{PCI\_IRDY}}$ to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates $\overline{\text{PCI\_IRDY}}$ to insert a wait cycle when it cannot accept data from the target.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional initiator ready, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates another PCI master is able to complete the current data phase of a transaction. Negated—If $\overline{\text{PCI\_FRAME}}$ is asserted, indicates a wait cycle from another master. If $\overline{\text{PCI\_FRAME}}$ is negated, indicates the PCI bus is idle.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	

Table 17-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_PAR	I/O	PCI parity. The PCI parity signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional PCI parity, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates odd parity across the PCI_AD[31:0] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals during address and data phases. Negated—Indicates even parity across the PCI_AD[31:0] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals during address and data phases.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional PCI parity, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases. Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
$\overline{\text{PCI\_PERR}}$	I/O	PCI parity error. The PCI parity error signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional PCI parity error, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller, acting as a PCI agent, detected a data parity error. (The PCI initiator drives $\overline{\text{PCI\_PERR}}$ on read operations; the PCI target drives $\overline{\text{PCI\_PERR}}$ on write operations.) Negated—Indicates no error.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional PCI parity error, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that another PCI agent detected a data parity error while this PCI controller was sourcing data (this PCI controller was acting as the PCI initiator during a write, or was acting as the PCI target during a read). Negated—Indicates no error.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
PCI_REQ[4:0]	I	PCI bus request. Input signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled, PCI_REQ[0] is an output. Note that PCI_REQ[n] is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the $\overline{\text{PCI\_REQ}}[n]$ input.	
		<b>State Meaning</b>	Asserted—Indicates that agent <i>n</i> is requesting control of the PCI bus to perform a transaction. Negated—Indicates that agent <i>n</i> does not require use of the PCI bus.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2

**Table 17-2. PCI Interface Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description	
$\overline{\text{PCI\_SERR}}$	I/O	PCI system error. The PCI system error signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional PCI system error, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that an address parity error, a target-abort (when this PCI controller is acting as the initiator), or some other system error (where the result is a catastrophic error) was detected. Negated—Indicates no error.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional PCI system error, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that a target (other than this PCI controller) has detected a catastrophic error. Negated—Indicates no error.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
$\overline{\text{PCI\_STOP}}$	I/O	Stop. The stop signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional stop, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller, acting as a PCI target, is requesting that the initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional stop, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that a target is requesting that the PCI initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	
$\overline{\text{PCI\_TRDY}}$	I/O	Target ready. Both an input and output signal on this PCI controller.	
	O	As outputs for the bidirectional target ready, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Indicates that this PCI controller, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, this PCI controller asserts $\overline{\text{PCI\_TRDY}}$ to indicate that valid data is present on the data bus. During a write, this PCI controller asserts $\overline{\text{PCI\_TRDY}}$ to indicate that it is prepared to accept data. Negated—Indicates that the PCI initiator needs to wait before this PCI controller, acting as a PCI target, can complete the current data phase. During a read, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot accept data from the initiator.
		<b>Timing</b>	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2
	I	As inputs for the bidirectional target ready, these signals operate as described below.	
		<b>State Meaning</b>	Asserted—Another PCI target is able to complete the current data phase of a transaction. Negated—Indicates a wait cycle from another target.
<b>Timing</b>		Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2	



**Table 17-2. PCI Interface Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description
PCI_CLK	I	PCI clock is an independent clock that may be used for the PCI interface. If used the PCI operation is asynchronous with respect to SYSCLK and the platform clock. In order to used this signal as the PCI clock source, it must be designated during POR configuration. See the reset chapter for POR details regarding clock selection as well as proper PCI frequency selection.
		<b>Timing</b> Assertion/Negation—See the device <i>Hardware Specification</i> for specific timing information.

## 17.3 Memory Map/Register Definitions

The PCI controller supports the following two types of registers:

- Memory-mapped registers—these registers control PCI address translation, PCI error management, and PCI configuration register access. These registers are described in [Section 17.3.1, “PCI Memory-Mapped Registers,”](#) and its subsections.
- PCI configuration registers contained within the PCI configuration header—these registers are specified by the PCI bus specification for every PCI device. These registers are described in [Section 17.3.2, “PCI Configuration Header,”](#) and its subsections.

### 17.3.1 PCI Memory-Mapped Registers

The PCI memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PCSRBAR on the PCI side) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI configuration data register, PCI\_CFG\_DATA) must only be accessed as 32-bit quantities.

[Table 17-3](#) lists the memory-mapped registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 17-3. PCI Memory-Mapped Register Map**

Offset	Register	Access	Reset	Section/page
<b>PCI Controller Memory-Mapped Registers—Block Base Address 0x0_8000</b>				
<b>PCI Configuration Access Registers</b>				
0x000	CFG_ADDR—PCI configuration address	R/W	0x0000_0000	<a href="#">17.3.1.1/17-14</a>

Table 17-3. PCI Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/page
0x004	CFG_DATA—PCI configuration data	R/W	0x0000_0000	17.3.1.1.2/17-15
0x008	INT_ACK—PCI interrupt acknowledge	R	0x0000_0000	17.3.1.1.3/17-15
0x00C– 0xBFC	Reserved	—	—	—
<b>PCI ATMU Registers—Outbound and Inbound</b>				
0xC00–0xC3C—Outbound Window 0 (default)				
0xC00	POTAR0—PCI outbound window 0 (default) translation address register	R/W	0x0000_0000	17.3.1.2.1/17-16
0xC04	POTEAR0—PCI outbound window 0 (default) translation extended address register	R/W	0x0000_0000	17.3.1.2.2/17-16
0xC08	Reserved	—	—	
0xC0C	Reserved	—	—	
0xC10	POWAR0—PCI outbound window 0 (default) attributes register	R/W	0x8004_401F	17.3.1.2.4/17-18
0xC14– 0xC1C	Reserved	—	—	
0xC20–0xC3C—Outbound Window 1				
0xC20	POTAR1—PCI outbound window 1 translation address register	R/W	0x0000_0000	17.3.1.2.1/17-16
0xC24	POTEAR1—PCI outbound window 1 translation extended address register	R/W	0x0000_0000	17.3.1.2.2/17-16
0xC28	POWBAR1—PCI outbound window 1 base address register	R/W	0x0000_0000	17.3.1.2.3/17-17
0xC2C	Reserved	—	—	
0xC30	POWAR1—PCI outbound window 1 attributes register	R/W	0x0000_0000	17.3.1.2.4/17-18
0xC34– 0xC3C	Reserved	—	—	
0xC40–0xC5C—Outbound Window 2				
0xC40	POTAR2—PCI outbound window 2 translation address register	R/W	0x0000_0000	17.3.1.2.1/17-16
0xC44	POTEAR2—PCI outbound window 2 translation extended address register	R/W	0x0000_0000	17.3.1.2.2/17-16
0xC48	POWBAR2—PCI outbound window 2 base address register	R/W	0x0000_0000	17.3.1.2.3/17-17
0xC4C	Reserved	—	—	
0xC50	POWAR2—PCI outbound window 2 attributes register	R/W	0x0000_0000	17.3.1.2.4/17-18
0xC54– 0xC5C	Reserved	—	—	
0xC60–0xC7C—Outbound Window 3				
0xC60	POTAR3—PCI outbound window 3 translation address register	R/W	0x0000_0000	17.3.1.2.1/17-16
0xC64	POTEAR3—PCI outbound window 3 translation extended address register	R/W	0x0000_0000	17.3.1.2.2/17-16
0xC68	POWBAR3—PCI outbound window 3 base address register	R/W	0x0000_0000	17.3.1.2.3/17-17
0xC6C	Reserved	—	—	
0xC70	POWAR3—PCI outbound window 3 attributes register	R/W	0x0000_0000	17.3.1.2.4/17-18

Table 17-3. PCI Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/page
0xC74– 0xC7C	Reserved	—	—	
0xC80–0xC9C—Outbound Window 4				
0xC80	POTAR4—PCI outbound window 4 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC84	POTEAR4—PCI outbound window 4 translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>
0xC88	POWBAR4—PCI outbound window 4 base address register	R/W	0x0000_0000	<a href="#">17.3.1.2.3/17-17</a>
0xC8C	Reserved	—	—	
0xC90	POWAR4—PCI outbound window 4 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.2.4/17-18</a>
0xC94– 0xD9C	Reserved	—	—	
0xDA0–0xDBC—Inbound Window 3				
0xDA0	PITAR3—PCI inbound window 3 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0xDA4	Reserved	—	—	
0xDA8	PIWBAR3—PCI inbound window 3 base address register	R/W	0x0000_0000	<a href="#">17.3.1.3.2/17-20</a>
0xDAC	PIWBEAR3—PCI inbound window 3 base extended address register	R/W	0x0000_0000	<a href="#">17.3.1.3.3/17-21</a>
0xDB0	PIWAR3—PCI inbound window 3 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0xDB4– 0xDBC	Reserved	—	—	
0xDC0–0xDDC—Inbound Window 2				
0xDC0	PITAR2—PCI inbound window 2 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0xDC4	Reserved	—	—	
0xDC8	PIWBAR2—PCI inbound window 2 base address register	R/W	0x0000_0000	<a href="#">17.3.1.3.2/17-20</a>
0xDCC	PIWBEAR2—PCI inbound window 2 base extended address register	R/W	0x0000_0000	<a href="#">17.3.1.3.3/17-21</a>
0xDD0	PIWAR2—PCI inbound window 2 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0xDD4– 0xDDC	Reserved	—	—	
0xDE0–0xDFC—Inbound Window 1				
0xDE0	PITAR1—PCI inbound window 1 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0xDE4	Reserved	—	—	
0xDE8	PIWBAR1—PCI inbound window 1 base address register	R/W	0x0000_0000	<a href="#">17.3.1.3.2/17-20</a>
0xDEC	Reserved	—	—	
0xDF0	PIWAR1—PCI inbound window 1 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0xDF4– 0xDFC	Reserved	—	—	
<b>PCI Error Management Registers</b>				
0xE00	ERR_DR—PCI error detect register	w1c	0x0000_0000	<a href="#">17.3.1.4.1/17-24</a>
0xE04	ERR_CAP_DR—PCI error capture disabled register	R/W	0x0000_0000	<a href="#">17.3.1.4.2/17-25</a>
0xE08	ERR_EN—PCI error enable register	R/W	0x0000_0000	<a href="#">17.3.1.4.3/17-26</a>

**Table 17-3. PCI Memory-Mapped Register Map (continued)**

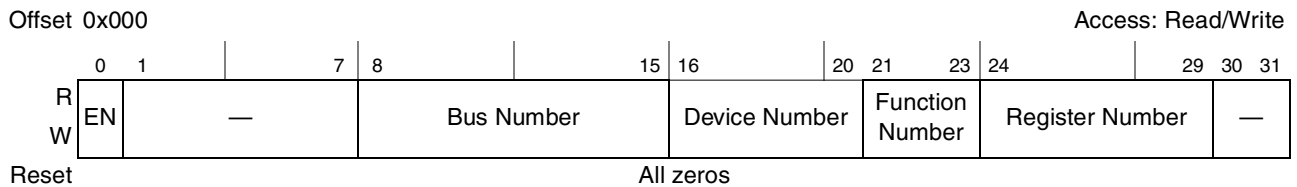
Offset	Register	Access	Reset	Section/page
0xE0C	ERR_ATTRIB—PCI error attributes capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.4/17-27</a>
0xE10	ERR_ADDR—PCI error address capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.5/17-28</a>
0xE14	ERR_EXT_ADDR—PCI error extended address capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.6/17-28</a>
0xE18	ERR_DL—PCI error data low capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.7/17-29</a>
0xE1C	ERR_DH—PCI error data high capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.8/17-29</a>
0xE20	GAS_TIMR—PCI gasket timer register	R/W	0x0100_3FFF	<a href="#">17.3.1.4.9/17-29</a>
0xE28– 0xEFC	Reserved	—	—	
0xF00– 0xFFC	Reserved for debug	—	—	

### 17.3.1.1 PCI Configuration Access Registers

The PCI configuration header, shown in [Figure 17-24](#) and [Figure 17-58](#), is accessed via an indirect method utilizing a pair of 32-bit memory-mapped access registers. For PCI, CFG\_ADDR is at offset 0x000 and CFG\_DATA is at offset 0x004.

#### 17.3.1.1.1 PCI Configuration Address Register (CFG\_ADDR)

The CFG\_ADDR register is shown in [Figure 17-3](#).



**Figure 17-3. PCI CFG\_ADDR Register**

[Table 17-4](#) describes the bit settings for the CFG\_ADDR register.

**Table 17-4. PCI CFG\_ADDR Field Descriptions**

Bits	Name	Description
0	Enable	Allow a PCI configuration access when PCI CFG_DATA is accessed
1–7	—	Reserved
8–15	Bus Number	PCI bus number to access
16–20	Device Number	Device number to access on specified bus
21–23	Function Number	Function to access within specified device
24–29	Register Number	32-bit register to access within specified device
30–31	—	Reserved, hardwired to logic 00

Bus number 0xb00 and device number 0b0\_0000 are used to configure the internal PCI configuration header of the PCI controller itself.

See [Section 17.4.2.11.2, “Host Accessing the PCI Configuration Space,”](#) and [Section 17.4.2.11.3, “Agent Accessing the PCI Configuration Space,”](#) for usage of PCI\_CFG\_ADDR.

### 17.3.1.1.2 PCI Configuration Data Register (CFG\_DATA)

The CFG\_DATA register is shown in [Figure 17-3](#).



**Figure 17-4. PCI\_CFG\_DATA Register**

[Table 17-5](#) describes the bit settings for the CFG\_DATA register

**Table 17-5. PCI\_CFG\_DATA Field Descriptions**

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI configuration cycle if the PCI_CFG_ADDR enable bit is set. If the enable bit is not set, a PCI I/O transaction is generated.

The CFG\_DATA register is a 4-byte window into the little-endian PCI configuration header data structure; therefore, byte addressing within the CFG\_DATA register uses little-endian convention. Note that CFG\_DATA may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

See [Section 17.4.2.11.2, “Host Accessing the PCI Configuration Space,”](#) and [Section 17.4.2.11.3, “Agent Accessing the PCI Configuration Space,”](#) for usage of CFG\_DATA.

### 17.3.1.1.3 PCI Interrupt Acknowledge Register (INT\_ACK)

An external PCI interrupt acknowledge transaction is generated by reading the INT\_ACK register. For PCI, INT\_ACK is at offset 0x008. INT\_ACK is shown in [Figure 17-5](#).



**Figure 17-5. PCI\_INT\_ACK Register**

[Table 17-6](#) describes the bit settings for the INT\_ACK register.

**Table 17-6. PCI\_INT\_ACK Field Descriptions**

Bits	Name	Description
0–31	Data	A read to this register generates a PCI interrupt acknowledge cycle.

### 17.3.1.2 PCI ATMU Outbound Registers

The outbound address translation and mapping unit controls the mapping of transactions from the internal platform address space to the external PCI address space. The outbound ATMU consists of four translation windows plus a default translation for transactions that do not hit in one of the four windows.

Each window contains a base address that points to the beginning of the window in the local address map, a translation address that specifies the high-order bits of the transaction in the external PCI address space, and a set of attributes including window size and external transaction type.

Each window must be aligned based on the granularity specified by the window size. If two outbound ATMU windows overlap in the local address space, the mapping of the lower numbered window has precedence over the higher numbered window.

Window 0 is the default window and is the only window enabled upon reset. The default outbound register set is used when a transaction misses in all of the other outbound windows.

#### 17.3.1.2.1 PCI Outbound Translation Address Registers (POTAR<sub>n</sub>)

The PCI outbound translation address registers (POTAR<sub>n</sub>) select the starting addresses in the PCI address space for hits in the PCI outbound windows. The translated address is created by concatenating the transaction offset to this translation address. The format of the POTAR<sub>n</sub> is shown in Figure 17-6.

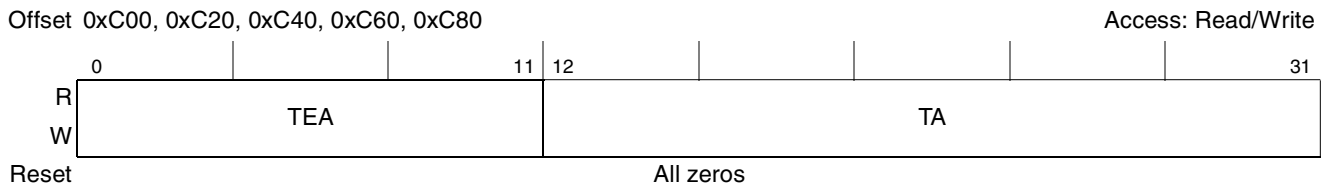


Figure 17-6. PCI Outbound Translation Address Registers (POTAR<sub>n</sub>)

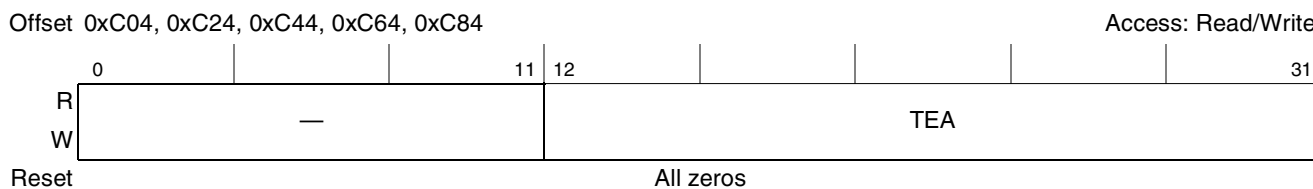
Table 17-7 describes the fields of the POTAR<sub>n</sub> registers.

Table 17-7. POTAR<sub>n</sub> Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. Represents bits [43:32] of a 64-bit PCI address (bit 0 is lsb).
12–31	TA	Translation address. Represents bits [31:12] of the PCI address. Based on the size of the window specified in POWAR <sub>n</sub> [OWS], the low-order bits of this field may be ignored.

#### 17.3.1.2.2 PCI Outbound Translation Extended Address Registers (POTEAR<sub>n</sub>)

The PCI outbound translation extended address registers (POTEAR<sub>n</sub>) contain the most significant bits of a 64-bit translation address. The format of POTEAR<sub>n</sub> is shown in Figure 17-7.



**Figure 17-7. PCI Outbound Translation Extended Address Registers (POTEAR $n$ )**

Table 17-8 describes the fields of the POTEAR $n$ .

**Table 17-8. POTEAR $n$  Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. Comprise bits [63:44] of the translation address.

### 17.3.1.2.3 PCI Outbound Window Base Address Registers (POWBAR $n$ )

The PCI outbound window base address registers (POWBAR $n$ ) point to the beginning of each translation window in the local 32-bit address space. Addresses for outbound transactions are compared to the appropriate bits in these registers, according to the sizes of the windows. If a transaction does not fall within one of these windows, the default translation and mapping is used. The default window is always enabled and used when the other windows miss.

Note that POWBAR $_0$  (for outbound ATMU window 0) is not used, because window 0 is the default window used when no other windows match. POWBAR $_0$  may be read from and written to, but the value is ignored.

The format of the POWBAR $n$  is shown in Figure 17-8.



**Figure 17-8. PCI Outbound Window Base Address Registers (POWBAR $n$ )**

Table 17-9 describes the field of the POWBAR $n$ .

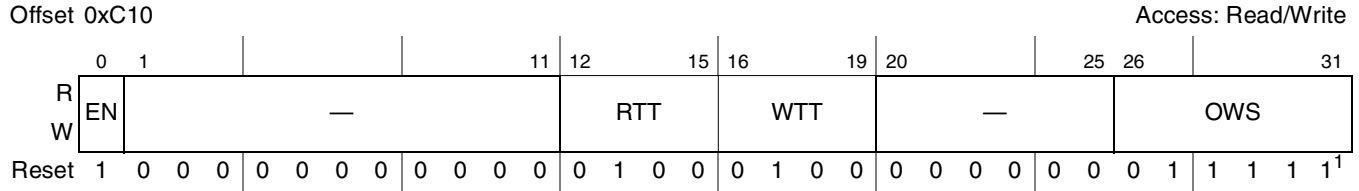
**Table 17-9. POWBAR $n$  Field Descriptions**

Bits	Name	Description
0–11	WBEA	Window base extended address. Bits 0–7 are reserved; bits 8–11 correspond to bits [0:3] of the (internal platform) base address. 0x000 – 0x00F are valid. 0x010 and greater are reserved.
12–31	WBA	Window base address. Source address which is the starting point for the outbound window. The window must be aligned based on the size selected in the window size bits. Corresponds to bits [4-35] of the (internal platform) base address.

### 17.3.1.2.4 PCI Outbound Window Attributes Registers (POWAR<sub>n</sub>)

The PCI outbound window attributes registers (POWAR<sub>n</sub>) define the window sizes to translate and other attributes for the translations. The minimum window size is 4 Kbytes. The maximum window size is 16 Gbytes.

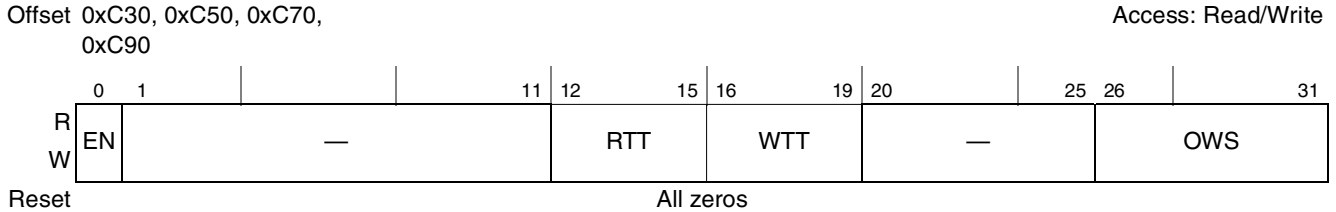
The default window attribute register, POWAR<sub>0</sub>, is shown in [Figure 17-9](#). Note that the fields for all of the POWAR<sub>n</sub> registers are the same, only the reset values are different.



**Figure 17-9. PCI Outbound Window 0 (Default) Attributes Register (POWAR<sub>0</sub>)**

<sup>1</sup> The default window is enabled, configured for memory read and memory write, and set to an OWS size of 4 Gbytes.

POWAR<sub>1</sub>–POWAR<sub>4</sub> are shown in [Figure 17-10](#).



**Figure 17-10. PCI Outbound Window 1–4 Attributes Registers (POWAR<sub>1</sub>–POWAR<sub>4</sub>)**

[Table 17-10](#) describes the fields for the POWAR<sub>n</sub> registers.

**Table 17-10. POWAR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0	EN	Enable. Enables this address translation
1–11	—	Reserved
12–15	RTT	Read transaction type to run on PCI 0000 Reserved ... 0011 Reserved 0100 Memory Read 0101 Reserved ... 0111 Reserved 1000 I/O Read 1001 Reserved ... 1111 Reserved



Table 17-10. POWAR<sub>n</sub> Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type to run on PCI 0000 Reserved ... 0011 Reserved 0100 Memory Write 0101 Reserved ... 0111 Reserved 1000 I/O Write 1001 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000Reserved ... 0010114-Kbyte window size 0011008-Kbyte window size ... 0111114-Gbyte window size 1000008-Gbyte window size 10000116-Gbyte window size 100010Reserved ... 111111Reserved The default POWAR register (0xC10) has an OWS value of 011111.

### 17.3.1.3 PCI ATMU Inbound Registers

The inbound address translation and mapping unit controls the mapping of transactions from the external PCI address space to the internal platform address space. The inbound ATMU is comprised of four windows—a configuration window and three general translation windows. The configuration window has higher priority than all other inbound ATMU windows and takes precedence over them if there is an overlap.

Each window contains the following:

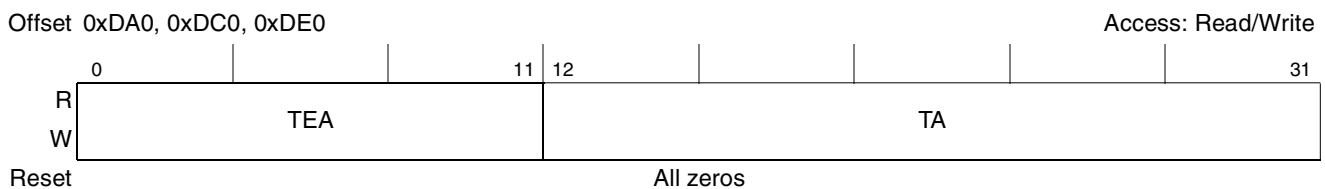
- A base address, which points to the beginning of the window in the external PCI address map. The base address of each window is also accessible by PCI configuration transactions as base address registers within the PCI configuration header, as shown in [Figure 17-26](#). The registers may be read or updated equivalently through the ATMU memory map or through PCI configuration transactions to the PCI configuration header.
- A translation address, which specifies the upper order bits of the transaction in the local address space.
- A set of attributes including window size and internal transaction attributes.

Each window’s base address and translation address must be aligned to the size of the window. If two general inbound ATMU windows overlap in the external PCI address space, the mappings of the lower numbered window are applied; PCSRBAR takes priority over any overlapping inbound ATMU window. In addition, if inbound ATMU windows are overlapped, the ATMU windows must not map to the same address with different sets of attributes (other than window size).

Note that PCSRBAR in the PCI configuration header acts as a fourth inbound window that translates a 1-Mbyte region of PCI space to the local configuration space pointed to by CCSRBAR. PCSRBAR can be accessed by PCI configuration cycles or by accessing the PCI configuration header through the PCI CFG\_ADDR and PCI\_CFG\_DATA registers. See [Section 17.3.1.1.1, “PCI Configuration Address Register \(CFG\\_ADDR\),”](#) [Section 17.3.1.1.2, “PCI Configuration Data Register \(CFG\\_DATA\),”](#) and [Section 17.3.2.11, “PCI Base Address Registers.”](#) All accesses to PCSRBAR have an automatic internal byte lane redirection from the little-endian PCI bus to the big-endian CCSRBAR configuration space.

### 17.3.1.3.1 PCI Inbound Translation Address Registers (PITAR<sub>n</sub>)

The PCI inbound translation address registers (PITAR<sub>n</sub>) points to the beginning of the local address space for the inbound window. The translated address is created by concatenating the transaction offset to this translation address. The format of the PITAR<sub>n</sub> is shown in [Figure 17-11](#).



**Figure 17-11. PCI Inbound Translation Address Registers (PITAR<sub>n</sub>)**

[Table 17-11](#) describes the fields of the PITAR<sub>n</sub> registers

**Table 17-11. PITAR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–11	TEA	Translation extended address. Bits 0–7 are reserved; bits 8–11 correspond to bits [0:3] of the local translation address. 0x000 – 0x00F are valid. 0x010 and greater are reserved.
12–31	TA	Translation address. Indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. TA corresponds to bits [4:23] of the 36-bit local translation address.

### 17.3.1.3.2 PCI Inbound Window Base Address Registers (PIWBAR<sub>n</sub>)

The PCI inbound window base address registers (PIWBAR<sub>n</sub>) select the PCI base address for the windows that are translated to the internal platform address space. Addresses for inbound transactions are compared to these windows. If a PCI transaction does not fall within one of these spaces, then the PCI interface does not assert DEVSEL. The PIWBAR<sub>n</sub> is shown in [Figure 17-12](#).



**Figure 17-12. PCI Inbound Window Base Address Registers**

Table 17-12 describes the fields of the PIWBAR $n$  registers.

**Table 17-12. PIWBAR Field Descriptions**

Bits	Name	Description
0–11	BEA	Base extended address. Corresponds to bits 43–32 of a 64-bit PCI base address.
12–31	BA	Base address. Corresponds to bits 31–12 of a PCI base address.

### 17.3.1.3.3 PCI Inbound Window Base Extended Address Registers (PIWBEAR $n$ )

The PCI inbound window base extended address registers (PIWBEAR $n$ ) contain the most-significant bits of a 64-bit base address. Note that inbound window 1 supports only a 32-bit base address and does not define an inbound window base extended address register. The PIWBEAR $n$  are shown in Figure 17-13.



**Figure 17-13. PCI Inbound Window Base Extended Address Registers (PIWBEAR $n$ )**

Table 17-13 describes the fields of the PIWBEAR $n$  registers.

**Table 17-13. PIWBEAR Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12–31	BEA	Base extended address. Corresponds to bits 63–44 of a 64-bit PCI base address.

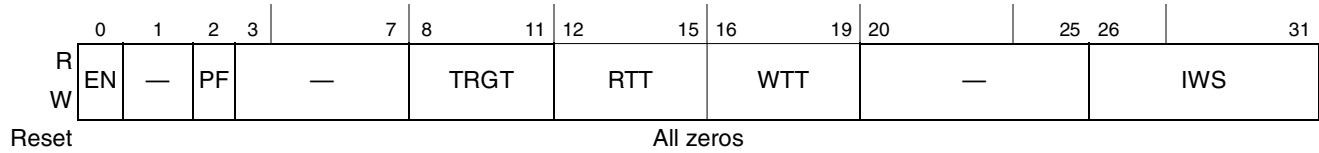
### 17.3.1.3.4 PCI Inbound Window Attributes Registers (PIWAR $n$ )

The PCI inbound window attributes registers (PIWAR $n$ ) define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The format of the PIWBAR $n$  is shown in Figure 17-14.

**PCI Bus Interface**

Offset 0xDB0, 0xDD0, 0xDF0

Access: Read/Write



**Figure 17-14. PCI Inbound Window Attributes Registers**

Table 17-14 describes the fields of the PIWAR<sub>n</sub> registers.

**Table 17-14. PIWAR<sub>n</sub> Field Descriptions**

Bits	Name	Description
0	EN	Enable. Enables this address translation
2	PF	Prefetchable. Indicates that the address space is prefetchable so that prefetching and streaming are attempted. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TRGT	Target interface. 0000 PCI Interface 0001 PCI Express interface 2 0010 PCI Express interface 1 0011 PCI Express interface 3 0100–1110 Reserved 1111 Local Memory (DDR SDRAM, Local Bus, SRAM) Note: If this field is set to an I/O port rather than local memory space, attributes for the external I/O transaction are assigned in an outbound ATMU of that I/O controller. Note also that it is illegal for a PCI or PCI Express interface to use itself as a target.
12–15	RTT	Read transaction type. Transaction type to run if access is a read. The field description differs subject to the transaction being targeted to I/O interface or to local memory. Following are the transaction type settings for reads to an I/O interface: 0000–0011 Reserved 0100 Read 0101–1111 Reserved Following are the transaction type settings for reads to local memory: 0000–0011 Reserved 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, unlock L2 cache line 1000–1111 Reserved

Table 17-14. PIWAR<sub>n</sub> Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Transaction type to run if access is a write. The field description differs subject to the transaction being targeted to an I/O interface or to local memory. Following are the transaction type settings for writes to an I/O interface: 0000–0011 Reserved 0100 Write 0101–1111 Reserved Following are the transaction type settings for writes to local memory: 0000–Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, allocate L2 cache line 0111 Write, allocate and lock L2 cache line 1000–1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window is 4 Kbytes. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved ... 111111 Reserved For configuration and run-time registers, the window size is fixed at 010011 1-Mbyte window size For register set 0, the window size is limited to 4 Gbytes or smaller.

### 17.3.1.4 PCI Error Management Registers

When a PCI error is detected, the appropriate error bit is set in the PCI error detect register. Subsequent errors set the appropriate error bits in the error detection registers, but relevant information (attributes, address, and data) is captured only for the first error. The PCI error detect register is a write-1-to-clear type register. That is, reading from this register occurs normally; however, write operations are different in that the bits can be cleared but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 25 and not affect any other bits in the register, the value 0x0000\_0040 is written to the register.

The error bit is set regardless of the state of the corresponding error enable bit in the PCI error enable register. The error enable bits are used to send or block the error reporting to the interrupt mechanism. The interrupt can be cleared by writing 0xFFFF\_FFFF to the PCI error detect register.

A master-abort condition during a configuration cycle is not necessarily an error. In this case, if relevant, the master abort error enable can be disabled to prevent the reporting of master-aborts during outbound configuration cycles. Master-aborts during configuration reads return 0xFFFF\_FFFF.

For an inbound configuration write transaction with a parity error, the device always updates the register access and generates the error interrupt if the interrupt enabled bit is set.

See Section 17.4.2.13, “PCI Error Functions,” for more detail on error handling.

### 17.3.1.4.1 PCI Error Detect Register (ERR\_DR)

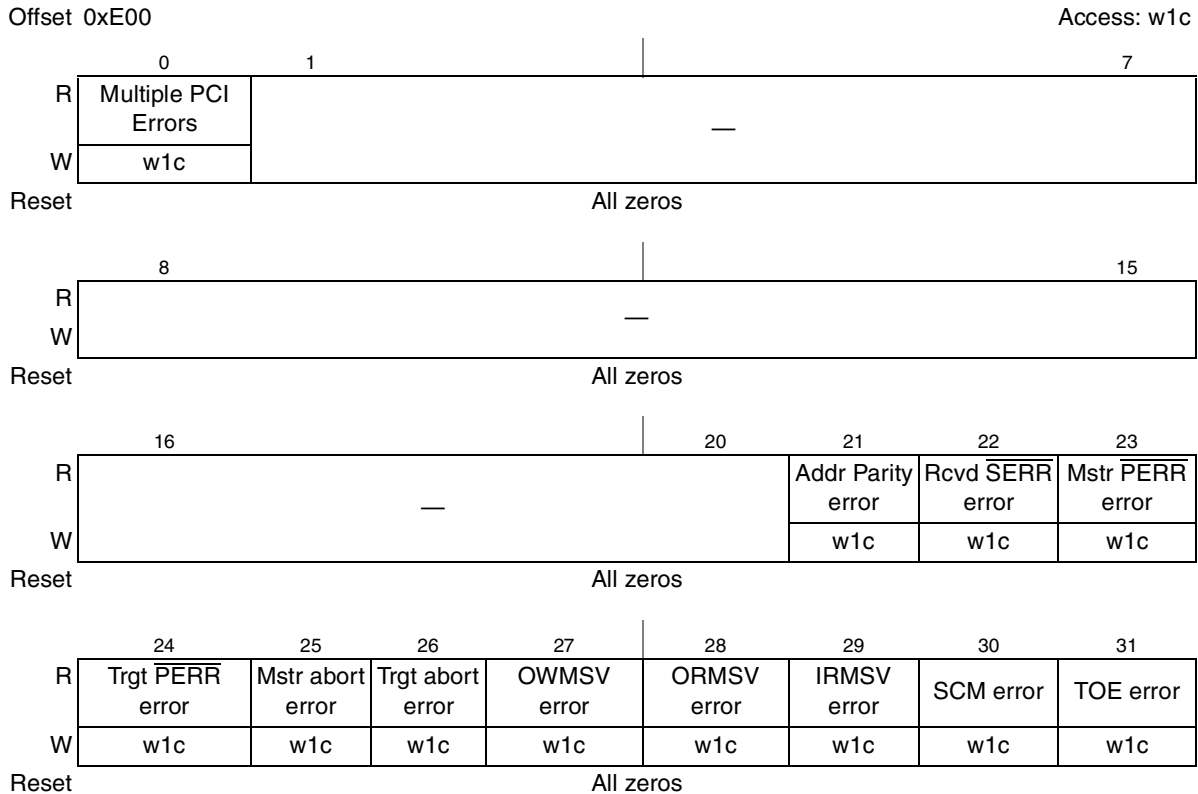


Figure 17-15. PCI Error Detect Register (ERR\_DR)

Table 17-15 describes ERR\_DR fields. Note that uncorrectable read errors may cause the assertion of *core\_fault\_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and an error occurs, the appropriate parity detect and master-abort bits in ERR\_DR must be cleared and the appropriate enable bits in ERR\_EN must be set to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 17-15. ERR\_DR Field Descriptions

Bits	Name	Description
0	Multiple PCI errors	0 Multiple PCI errors of the same type were not detected (write-1-to-clear) 1 Multiple PCI errors of the same type were detected
1–20	—	Reserved
21	Addr Parity error	Address parity error (write-1-to-clear)
22	Rcvd $\overline{\text{SERR}}$ error	Received $\overline{\text{SERR}}$ error (write-1-to-clear)
23	Mstr $\overline{\text{PERR}}$ error	Master $\overline{\text{PERR}}$ error (write-1-to-clear)

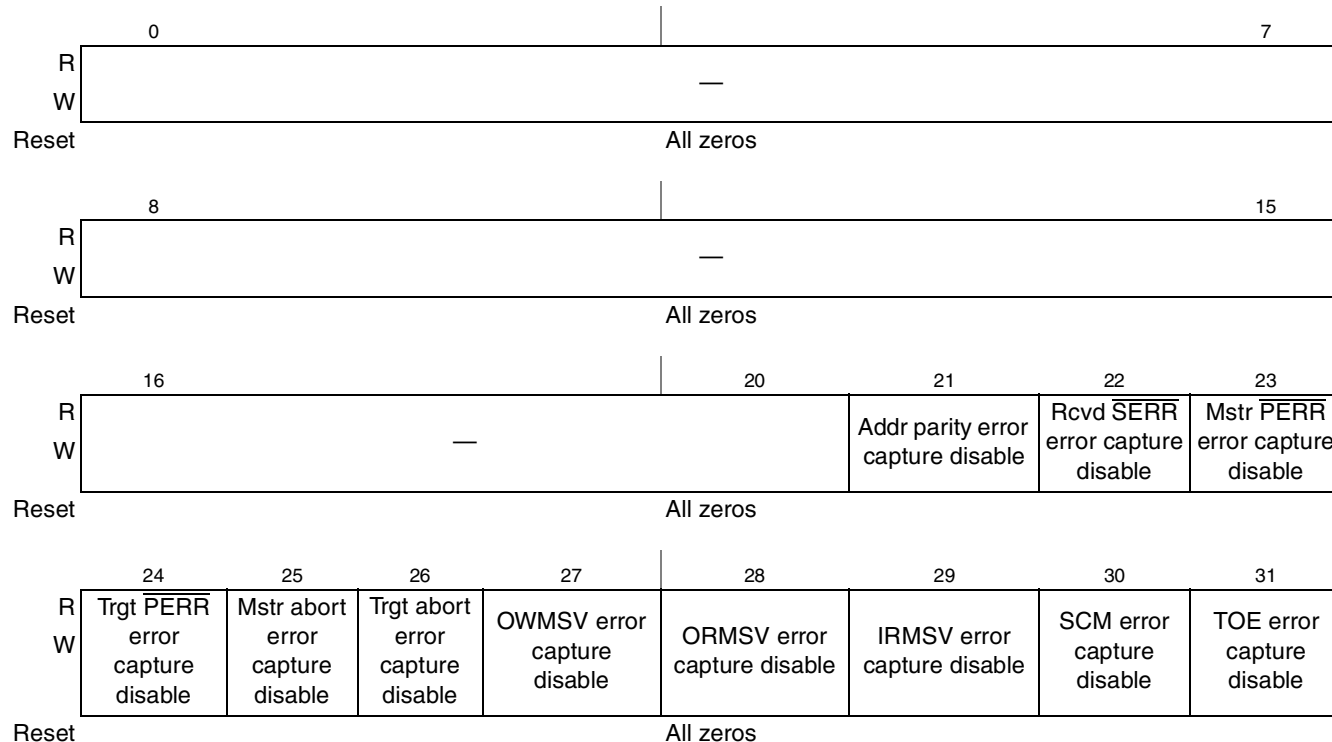
**Table 17-15. ERR\_DR Field Descriptions (continued)**

Bits	Name	Description
24	Trgt $\overline{\text{PERR}}$ error	Target $\overline{\text{PERR}}$ error (write-1-to-clear)
25	Mstr abort error	Master abort error (write-1-to-clear)
26	Trgt abort error	Target abort error (write-1-to-clear)
27	OWMSV error	Outbound write memory space violation error (write-1-to-clear)
28	ORMSV error	Outbound read memory space violation error (write-1-to-clear)
29	IRMSV error	Inbound read memory space violation error (write-1-to-clear)
30	SCM error	Split completion message error (write-1-to-clear)
31	TOE error	Time-out error (write-1-to-clear)

**17.3.1.4.2 PCI Error Capture Disable Register (ERR\_CAP\_DR)**

Offset 0xE04

Access: Read/Write



**Figure 17-16. PCI Error Capture Disable Register (ERR\_CAP\_DR)**

**Table 17-16. ERR\_CAP\_DR Field Descriptions**

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error capture disable	Disable capture for address parity errors
22	Rcvd $\overline{\text{SERR}}$ error capture disable	Disable capture for received $\overline{\text{SERR}}$ errors

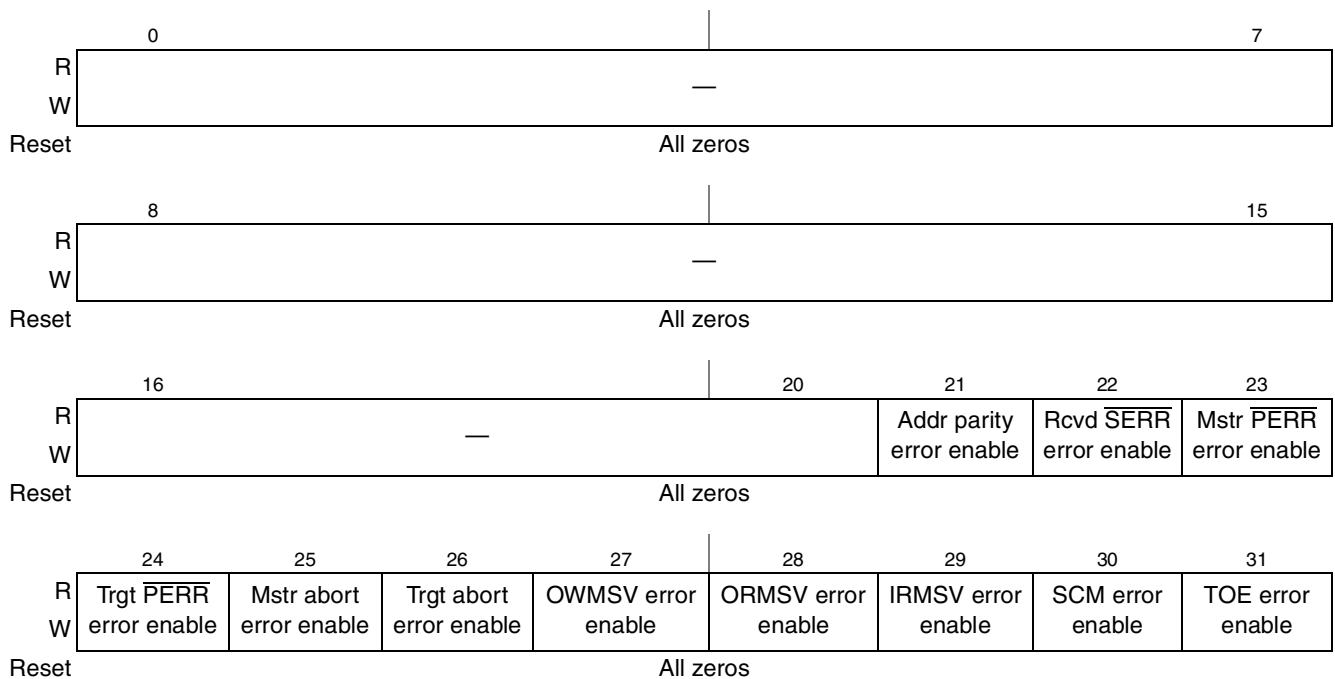
**Table 17-16. ERR\_CAP\_DR Field Descriptions (continued)**

Bits	Name	Description
23	Mstr $\overline{\text{PERR}}$ error capture disable	Disable capture for master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error capture disable	Disable capture for target $\overline{\text{PERR}}$ errors
25	Mstr abort error capture disable	Disable capture for master abort errors
26	Trgt abort error capture disable	Disable capture for target abort errors
27	OWMSV error capture disable	Disable capture for outbound write memory space violation errors
28	ORMSV error capture disable	Disable capture for outbound read memory space violation errors
29	IRMSV error capture disable	Disable capture for inbound read memory space violation errors
30	SCM error capture disable	Disable capture for split completion message errors
31	TOE error capture disable	Disable capture for time-out errors

**17.3.1.4.3 PCI Error Enable Register (ERR\_EN)**

Offset 0xE08

Access: Read/Write



**Figure 17-17. PCI Error Enable Register (ERR\_EN)**

Table 17-17 describes ERR\_EN fields. Note that uncorrectable read errors may cause the assertion of *core\_fault\_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, the appropriate parity detect and master-abort bits in ERR\_DR must be cleared and the appropriate enable bits in ERR\_EN must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”



Table 17-17. ERR\_EN Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error enable	Enable reporting address parity errors
22	Rcvd $\overline{\text{SERR}}$ error enable	Enable reporting received $\overline{\text{SERR}}$ errors
23	Mstr $\overline{\text{PERR}}$ error enable	Enable reporting master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error enable	Enable reporting target $\overline{\text{PERR}}$ errors
25	Mstr abort error enable	Enable reporting master abort errors
26	Trgt abort error enable	Enable reporting target abort errors
27	OWMSV error enable	Enable reporting outbound write memory space violation errors
28	ORMSV error enable	Enable reporting outbound read memory space violation errors
29	IRMSV error enable	Enable reporting inbound read memory space violation errors
30	SCM error enable	Enable reporting split completion message errors
31	TOE error enable	Enable reporting time-out errors

#### 17.3.1.4.4 PCI Error Attributes Capture Register (ERR\_ATTRIB)

Offset 0xE0C

Access: Read/Write

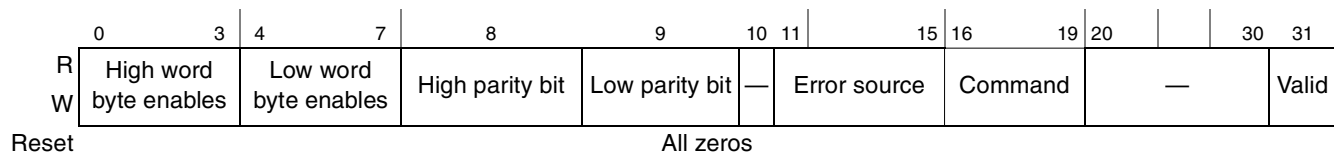


Figure 17-18. PCI Error Attributes Capture Register (ERR\_ATTRIB)

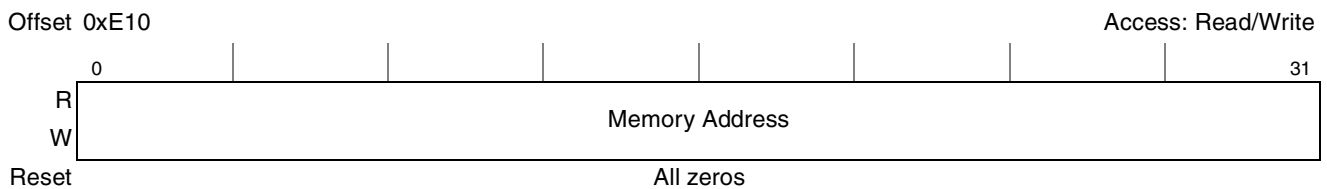
Table 17-18. ERR\_ATTRIB Field Descriptions

Bits	Name	Description
0–3	High word byte enables	PCI byte enables for most significant word of the double word
4–7	Low word byte enables	PCI byte enables for least significant word of the double word
8	High parity bit	Parity bit for most significant PCI bus data word (only valid for 64-bit PCI bus)
9	Low parity bit	Parity bit for least significant PCI bus data word
10	—	Reserved

**Table 17-18. ERR\_ATTRIB Field Descriptions (continued)**

Bits	Name	Description
11–15	Error source	The source of the PCI transaction 00000 PCI 00001 PCI Express 2 00010 PCI Express 1 00011 PCI Express 3 00100 Local bus controller 00101–01111 Reserved 10000 e500 core (instruction) 10001 e500 core (data) 10010 Reserved 10011 Reserved 10100 Reserved 10101 DMA 10110 Reserved 10111 Reserved 11000 eTSEC1 11001 Reserved 11010 eTSEC3 11011–11111 Reserved
16–19	Command	PCI command
20–30	—	Reserved
31	Valid info	The PCI bus capture registers contain valid information

**17.3.1.4.5 PCI Error Address Capture Register (ERR\_ADDR)**

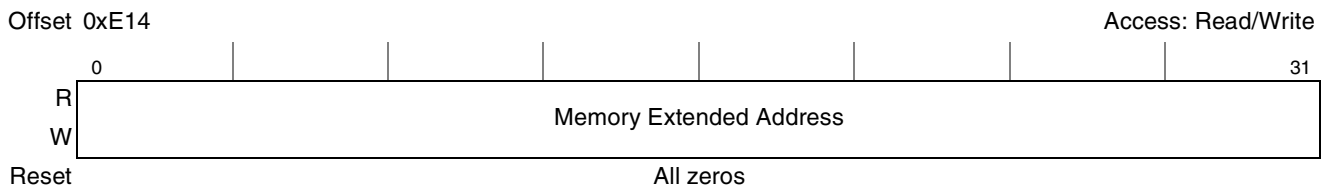


**Figure 17-19. PCI Error Address Capture Register (ERR\_ADDR)**

**Table 17-19. ERR\_ADDR Field Descriptions**

Bits	Name	Description
0–31	Memory address	Memory transaction address

**17.3.1.4.6 PCI Error Extended Address Capture Register (ERR\_EXT\_ADDR)**



**Figure 17-20. PCI Error Extended Address Capture Register (ERR\_EXT\_ADDR)**

**Table 17-20. ERR\_EXT\_ADDR Field Descriptions**

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

### 17.3.1.4.7 PCI Error Data Low Capture Register (ERR\_DL)



Figure 17-21. PCI Error Data Low Capture Register (ERR\_DL)

Table 17-21. ERR\_DL Field Description

Bits	Name	Description
0–31	Data low	Least significant PCI bus data word

### 17.3.1.4.8 PCI Error Data High Capture Register (ERR\_DH)



Figure 17-22. PCI Error Data High Capture Register (ERR\_DH)

Table 17-22. ERR\_DH Field Description

Bits	Name	Description
0–31	Data high	Most significant PCI bus data word (only valid with 64-bit PCI bus)

### 17.3.1.4.9 PCI Gasket Timer Register (GAS\_TIMR)

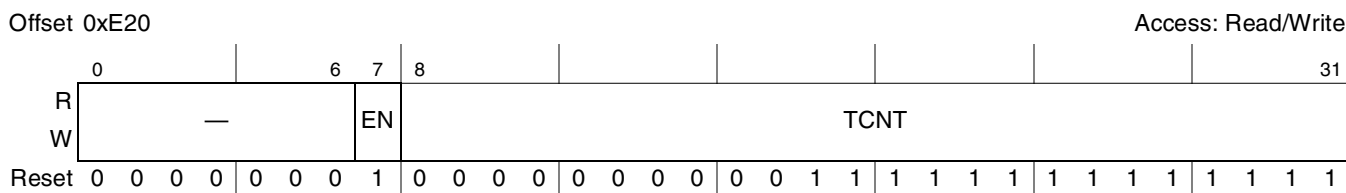


Figure 17-23. PCI Gasket Timer Register (GAS\_TIMR)

Table 17-23. GAS\_TIMR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	EN	Gasket timer enable. 0 gasket timer is disabled. 1 gasket timer is enabled.
8–31	TCNT	Number of system clocks to purge a non-prefetchable inbound read buffer

### 17.3.2 PCI Configuration Header

The *PCI Local Bus Specification* defines the configuration registers contained within the PCI configuration header from 0x00 through 0x3F. Figure 17-24 lists the common PCI configuration header as implemented by the device.

				Address Offset (Hex)	
Device ID				Vendor ID	00
PCI Bus Status		PCI Bus Command		04	
Bus Base Class Code	Subclass Code	Bus Programming Interface	Revision ID	08	
BIST Control	Header Type	Bus Latency Timer	Bus Cache Line Size	0C	
PCI Configuration and Status Register Base Address Register (PCSRBAR)				10	
32-Bit Memory Base Address Register				14	
64-Bit Low Memory Base Address Register				18	
64-Bit High Memory Base Address Register				1C	
64-Bit Low Memory Base Address Register				20	
64-Bit High Memory Base Address Register				24	
				28	
Subsystem ID		Subsystem Vendor ID		2C	
				30	
			PCI Bus Capability Pointer	34	
				38	
PCI Bus MAX_LAT	PCI Bus MIN_GNT	PCI Bus Interrupt Pin	PCI Bus Interrupt Line	3C	
				40	
PCI Bus Arbiter Configuration		PCI Bus Function		44	

Figure 17-24. Common PCI Configuration Header

Table 17-49 in Section 17.4.2.11.1, “PCI Configuration Space Header,” provides a summary of the PCI configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

#### 17.3.2.1 PCI Vendor ID Register—Offset 0x00

The PCI vendor ID register, shown in Figure 17-25, is used to identify the manufacturer of the part.

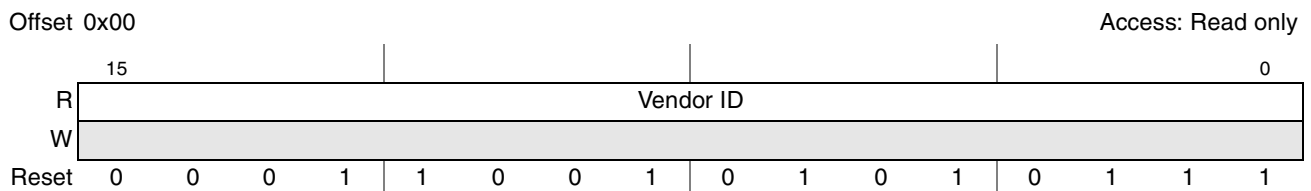


Figure 17-25. PCI Vendor ID Register

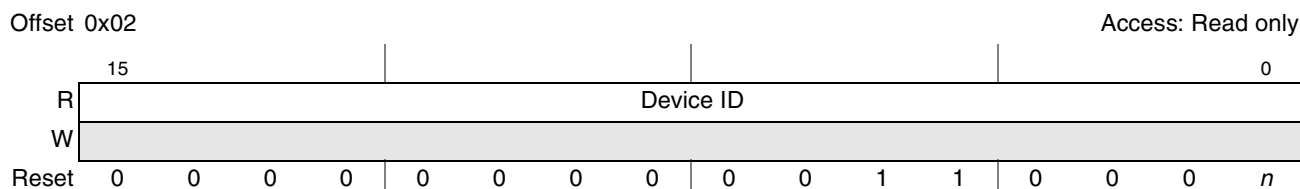
Table 17-24 describes PCI vendor ID register fields.

**Table 17-24. PCI Vendor ID Register Field Description**

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

### 17.3.2.2 PCI Device ID Register—Offset 0x02

The PCI device ID register, shown in Figure 17-26, is used to identify the device.



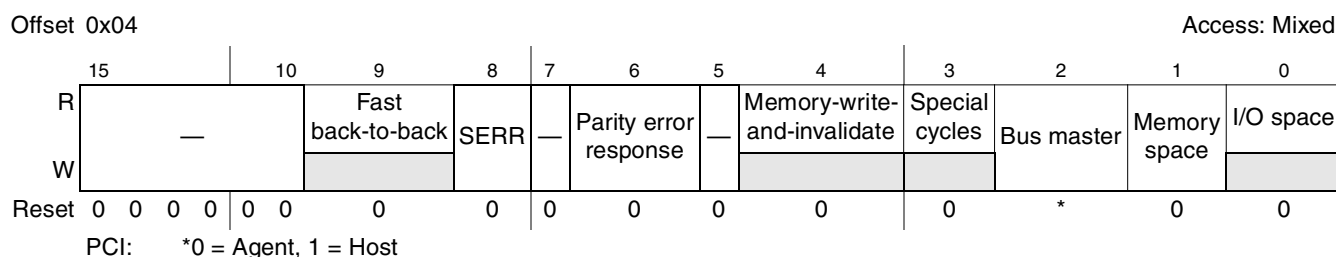
**Figure 17-26. PCI Device ID Register**

**Table 17-25. PCI Device ID Register Field Description**

Bits	Name	Description
15–0	Device ID	0x0030 MPC8533E (with security) 0x0031 MPC8533 (without security)

### 17.3.2.3 PCI Bus Command Register—Offset 0x04

The 2-byte PCI bus command register provides control over the ability to generate and respond to PCI cycles. Table 17-26 describes the bits of the PCI bus command register.



**Figure 17-27. PCI Bus Command Register**

**Table 17-26. PCI Bus Command Register Field Descriptions**

Bits	Name	Description
15–10	—	Reserved
9	Fast back-to-back	Hard-wired to 0, indicating that this PCI controller (as a master) does not run fast back-to-back transactions.

**Table 17-26. PCI Bus Command Register Field Descriptions (continued)**

Bits	Name	Description
8	SERR	Controls the $\overline{\text{PCI\_SERR}}$ driver of this PCI controller. This bit (and bit 6) must be set to report address parity errors. 0 Disables the $\overline{\text{PCI\_SERR}}$ driver 1 Enables the $\overline{\text{PCI\_SERR}}$ driver
7	—	Reserved
6	Parity error response	Controls whether this PCI controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Parity errors cause the appropriate bit in the PCI status register to be set. However, note that errors are reported based on the values set in the PCI error enable and detection registers.
5	—	Reserved
4	Memory-write-and-invalidate	Hard-wired to 0, indicating that this PCI controller, acting as a master, can not generate the memory-write-and-invalidate command.
3	Special-cycles	Hard-wired to 0, indicating that this PCI controller (as a target) ignores all special-cycle commands.
2	Bus master	Indicates whether this PCI controller is configured as a master. This indicates the setting of the host/agent configuration input signal at power-on reset. 0 Disables the ability to generate PCI accesses 1 Enables this PCI controller to behave as a PCI bus master (Host)
1	Memory space	Controls whether this PCI controller (as a target) responds to memory accesses. 0 This PCI controller does not respond to PCI memory space accesses. 1 This PCI controller (as a target) responds to PCI memory space accesses.
0	I/O space	Hard-wired to 0, indicating that this PCI controller (as a target) does not respond to PCI I/O space accesses.

### 17.3.2.4 PCI Bus Status Register—Offset 0x06

The 2-byte PCI bus status register is used to record status information for PCI bus-related events. The definition of each bit is given in [Table 17-27](#). Only 2-byte accesses to address offset 0x06 are allowed.

Reads to this register behave normally. Writes are slightly different in that bits can be cleared, but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 14 without affecting any other bits in the register, write the value 0b0100\_0000\_0000\_0000 to the register.

Offset 0x06

Access: Mixed

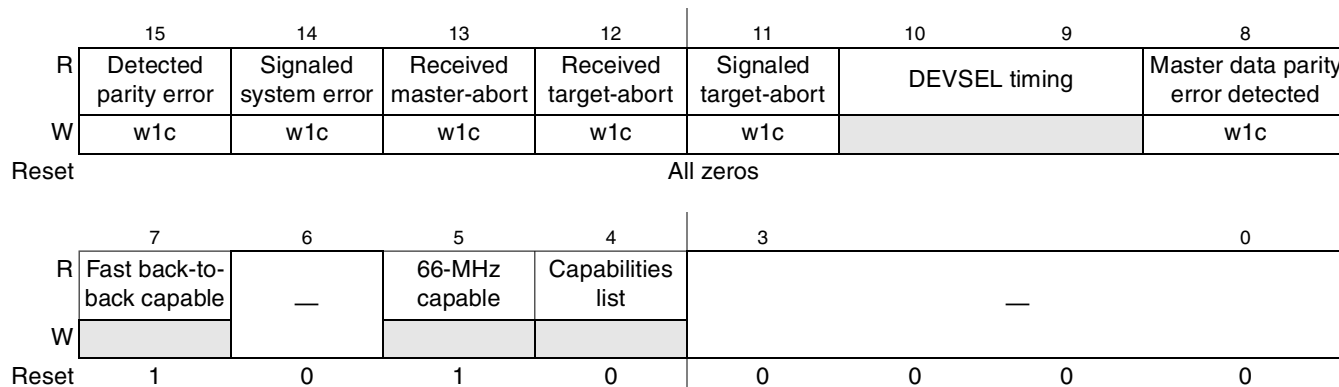


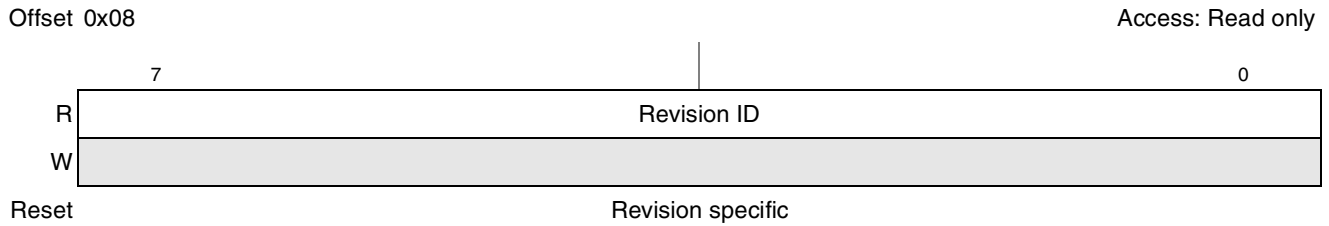
Figure 17-28. PCI Bus Status Register

Table 17-27. PCI Bus Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error	Set whenever this PCI controller detects a PCI parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI bus command register).
14	Signaled system error	Set whenever this PCI controller asserts $\overline{\text{PCI\_SERR}}$ .
13	Received master-abort	Set whenever this PCI controller, acting as the PCI master, terminates a transaction (except for a special-cycle) using master-abort.
12	Received target-abort	Set whenever a PCI transaction initiated by this PCI controller (excluding a special-cycle) is terminated by a target-abort.
11	Signaled target-abort	Set whenever this PCI controller, acting as the PCI target, issues a target-abort to a PCI master.
10–9	DEVSEL timing	Hard-wired to 0b00, indicating that this PCI controller uses fast device select timing.
8	Master data parity error detected	Set upon detecting a data parity error. Three conditions must be met for this bit to be set: <ul style="list-style-type: none"> <li>• This PCI controller detected a parity error.</li> <li>• This PCI controller was acting as the bus master for the operation in which the error occurred.</li> <li>• Bit 6 in the PCI bus command register was set.</li> </ul>
7	Fast back-to-back capable	Hard-wired to 1, indicating that this PCI controller (as a target) is capable of accepting fast back-to-back transactions.
6	—	Reserved
5	66-MHz capable	Read-only bit indicates that this PCI controller is capable of 66 MHz PCI bus operation.
4	Capabilities List	Hard-wired to 0
3–0	—	Reserved

### 17.3.2.5 PCI Revision ID Register—Offset 0x08

The PCI revision ID register is used to identify the revision of the part.



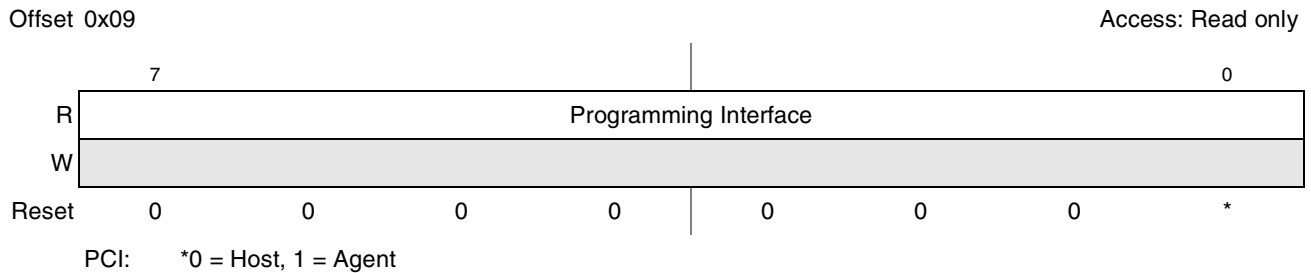
**Figure 17-29. PCI Revision ID Register**

**Table 17-28. PCI Revision ID Register Field Descriptions**

Bits	Name	Description
7-0	Revision ID	Revision specific

### 17.3.2.6 PCI Bus Programming Interface Register—Offset 0x09

Table 17-29 describes the PCI bus programming interface register (PIR).



**Figure 17-30. PCI Bus Programming Interface Register**

**Table 17-29. PCI Bus Programming Interface Register Field Description**

Bits	Name	Description
7-0	Programming Interface	0x00 When the PCI controller is configured as host bridge 0x01 When the PCI controller is configured as an agent device



### 17.3.2.7 PCI Subclass Code Register—Offset 0x0A

Table 17-31 describes the PCI subclass code register (PSCR).

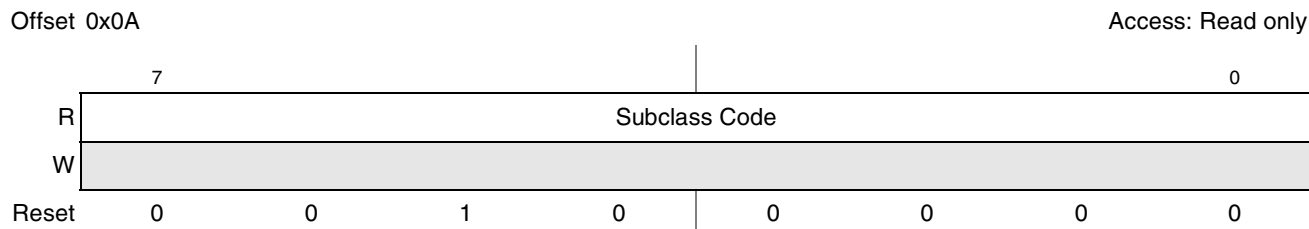


Figure 17-31. PCI Subclass Code Register

Table 17-30. PCI Subclass Code Register Field Description

Bits	Name	Description
7-0	Subclass Code	PowerPC—0x20

### 17.3.2.8 PCI Bus Base Class Code Register—Offset 0x0B

Table 17-31 describes the PCI bus base class code register (PBCCR).

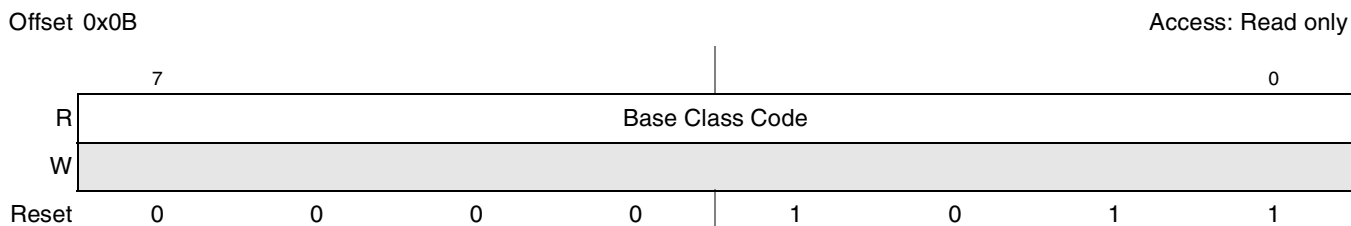


Figure 17-32. PCI Bus Base Class Code Register

Table 17-31. PCI Bus Base Class Code Register Field Description

Bits	Name	Description
7-0	Base Class Code	Processor—0x0B

### 17.3.2.9 PCI Bus Cache Line Size Register—Offset 0x0C

Table 17-32 describes the PCI bus cache line size register (PCLSR).

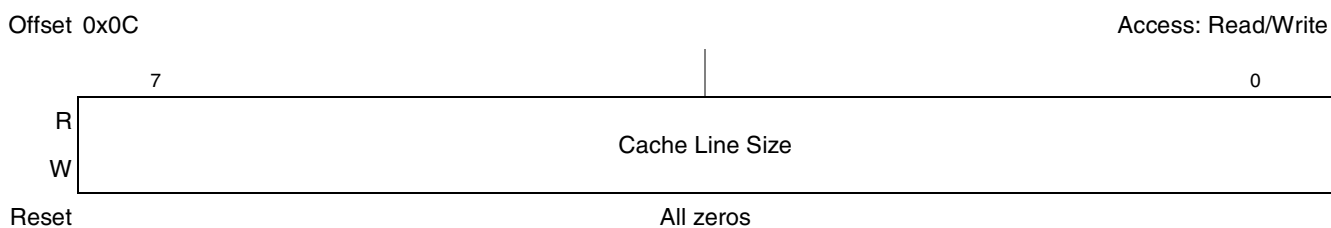


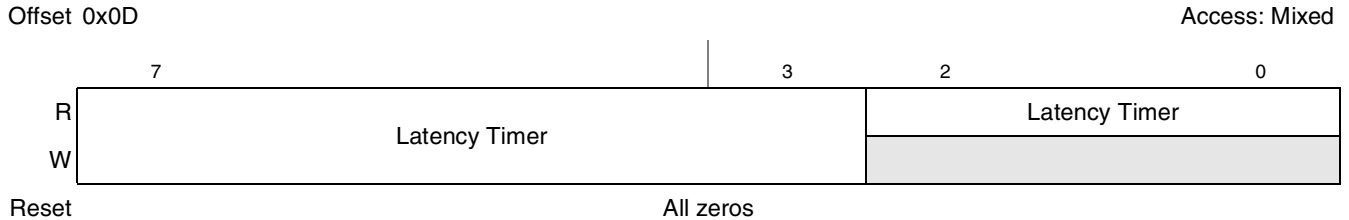
Figure 17-33. PCI Bus Cache Line Size Register

**Table 17-32. PCI Bus Cache Line Size Register Field Descriptions**

Bits	Name	Description
7–0	Cache Line Size	Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). PCLSR is read-write; however, for PCI operation an attempt to program this register to any value other than 0x8 results in clearing it.

### 17.3.2.10 PCI Bus Latency Timer Register—0x0D

Table 17-33 describes the PCI latency timer register (PLTR).



**Figure 17-34. PCI Bus Latency Timer Register**

**Table 17-33. PCI Bus Latency Timer Register Field Descriptions**

Bits	Name	Description
7–3	Latency Timer	The maximum number of PCI clocks that the device, when mastering a transaction, holds the bus after PCI bus grant has been negated. The value is in PCI clocks. The PCI 2.2 specification gives rules by which the PCI bus interface unit completes transactions when the timer has expired.
2–0	Latency Timer	Read-only bits. The minimum latency timer value when set is 8 PCI clocks.

### 17.3.2.11 PCI Base Address Registers

A PCI base address register points to the beginnings of each address range to which the device responds by asserting PCI\_DEVSEL. The base address register (BAR) at offset 0x10 is a fixed 1-Mbyte window that is automatically translated to the local configuration, control, and status registers address space.

The other base address registers are aliases (with differing format) of the PCI inbound ATMU windows; see Section 17.3.1.3, “PCI ATMU Inbound Registers.” The 32-bit base address register at offset 0x14 corresponds to inbound ATMU window 1; the 64-bit base address registers at offsets 0x18 and 0x20 correspond to inbound ATMU windows 2 and 3. If one of these registers is written, the corresponding ATMU register is also updated; if a PCI inbound ATMU register is written, the corresponding BAR is also updated. If one of these registers is read, the corresponding size of ATMU is returned on the PCI bus providing valid window size in the Inbound ATMU window attributes register.

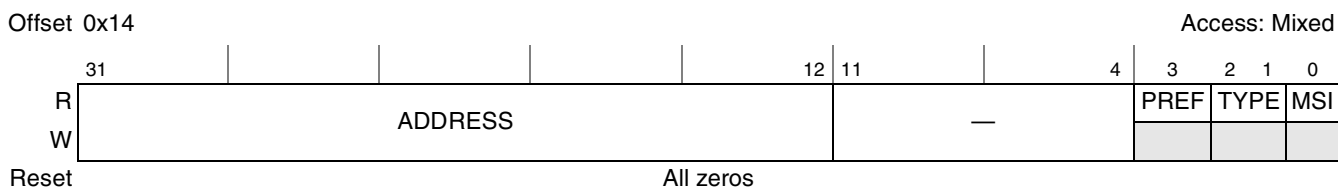
Note that PCSRBAR cannot be updated through the inbound ATMU registers.



**Figure 17-35. PCI Configuration and Status Register Base Address Register (PCSRBAR)**

**Table 17-34. PCSRBAR Field Descriptions**

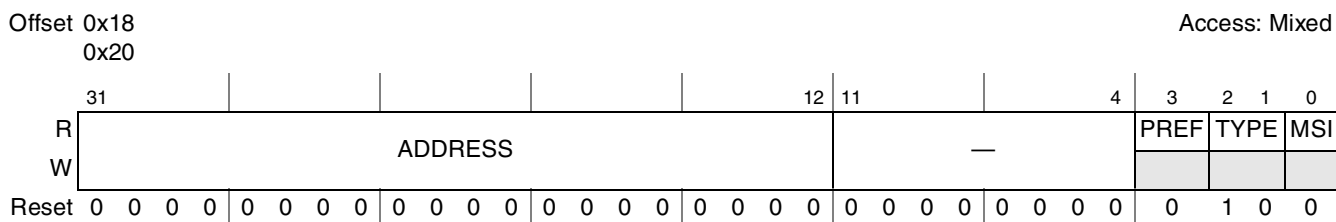
Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration/run-time window resides at. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MSI	Memory space indicator



**Figure 17-36. 32-Bit Memory Base Address Register**

**Table 17-35. 32-Bit Memory Base Address Register Field Descriptions**

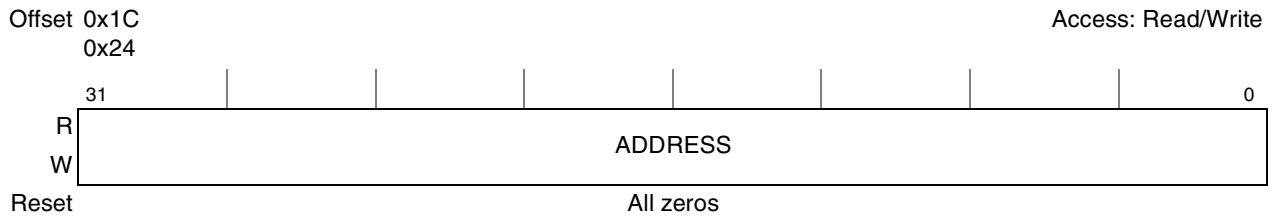
Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MSI	Memory space indicator.



**Figure 17-37. 64-Bit Low Memory Base Address Register**

**Table 17-36. 64-Bit Low Memory Base Address Register Field Descriptions**

Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type. 10 Locate anywhere in 64-bit address space.
0	MSI	Memory space indicator



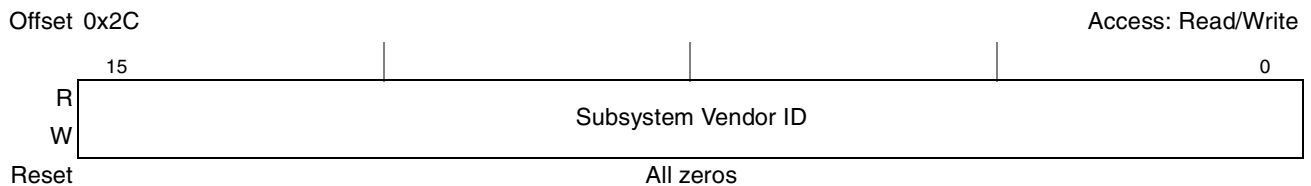
**Figure 17-38. 64-Bit High Memory Base Address Register**

**Table 17-37. Bit Setting for 64-Bit High Memory Base Address Register**

Bits	Name	Description
31–0	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows. If no access to local memory is to be permitted by external masters then all bits are programmed.

### 17.3.2.12 PCI Subsystem Vendor ID Register

The PCI subsystem vendor ID register is used to identify the subsystem.



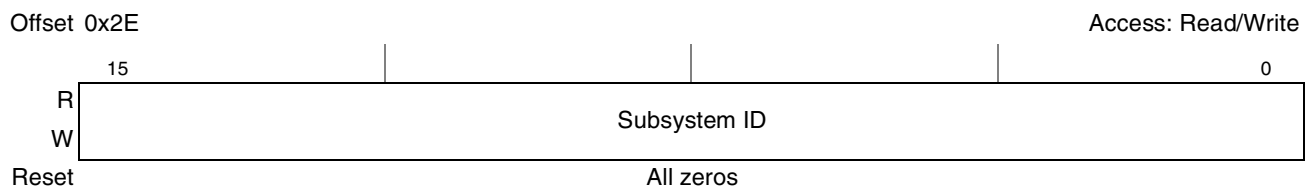
**Figure 17-39. PCI Subsystem Vendor ID Register**

**Table 17-38. PCI Subsystem Vendor ID Register Field Description**

Bits	Name	Description
15–0	Subsystem Vendor ID	0x0000

### 17.3.2.13 PCI Subsystem ID Register

The PCI subsystem ID register is used to identify the subsystem.



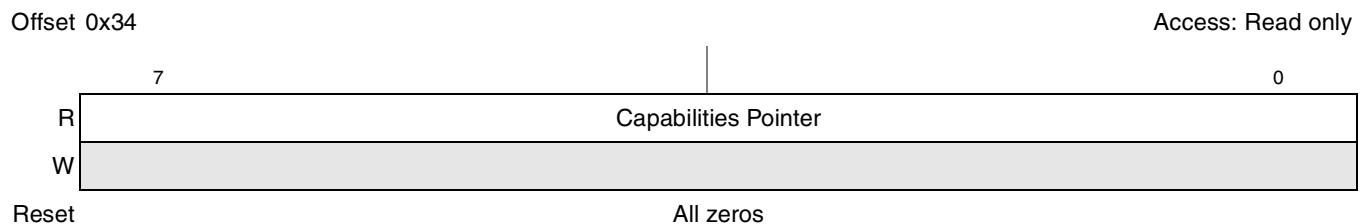
**Figure 17-40. PCI Subsystem ID Register**

**Table 17-39. PCI Subsystem ID Register Field Description**

Bits	Name	Description
15–0	Subsystem ID	0x0000

### 17.3.2.14 PCI Bus Capabilities Pointer Register

The PCI bus capabilities pointer identifies additional functionality supported by the device.

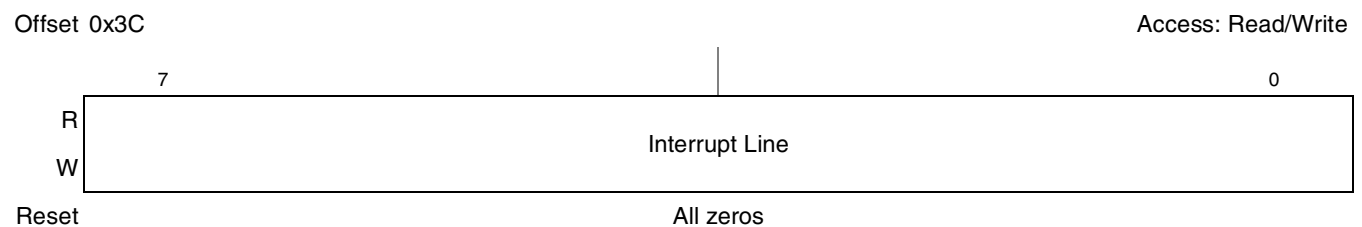


**Figure 17-41. PCI Bus Capabilities Pointer Register**

**Table 17-40. PCI Bus Capabilities Pointer Register Field Description**

Bits	Name	Description
7–0	Capabilities Pointer	No additional capabilities

### 17.3.2.15 PCI Bus Interrupt Line Register



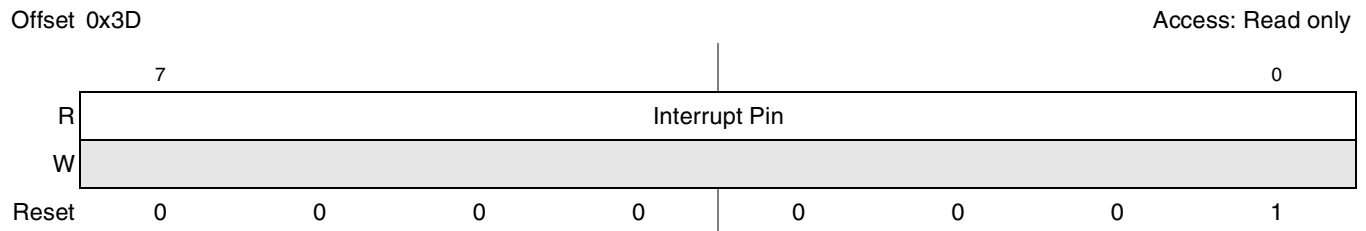
**Figure 17-42. PCI Bus Interrupt Line Register**

**Table 17-41. PCI Bus Interrupt Line Register Field Description**

Bits	Name	Description
7-0	Interrupt Line	Used to communicate interrupt line routing information.

### 17.3.2.16 PCI Bus Interrupt Pin Register

The programmable interrupt controller (PIC) has 12 general purpose interrupt request inputs (IRQ[0:11]) and an interrupt output, `IRQ_OUT` (active low, level sensitive), to which all external and most internal interrupt sources (including PCI) can be routed. `IRQ_OUT` is mapped to `PCI_INTA` as a default. Note that this device does not respond to `INTACK` or special cycle commands on the PCI interfaces.

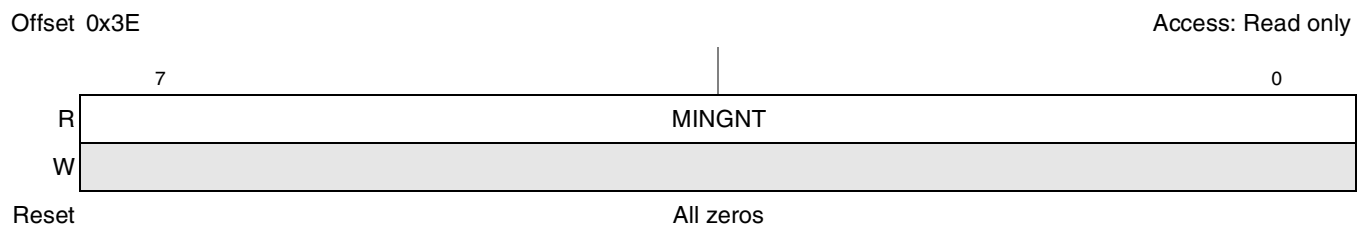


**Figure 17-43. PCI Bus Interrupt Pin Register**

**Table 17-42. PCI Bus Interrupt Pin Register Field Description**

Bits	Name	Description
7-0	Interrupt pin	PCI_INTA pin selected

### 17.3.2.17 PCI Bus Minimum Grant Register (MIN\_GNT)



**Figure 17-44. PCI Bus Minimum Grant Register (MIN\_GNT)**

**Table 17-43. PCI Bus Minimum Grant Register Field Description**

Bits	Name	Description
7-0	MINGNT	Specifies the length of the device's burst period (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers.)

### 17.3.2.18 PCI Bus Maximum Latency Register (MAX\_LAT)

Offset 0x3F

Access: Read Only

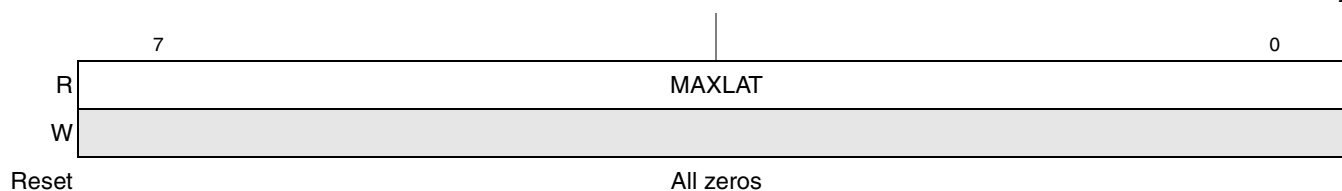


Figure 17-45. PCI Bus Maximum Latency Register (MAX\_LAT)

Table 17-44. PCI Bus Maximum Latency Register Field Description

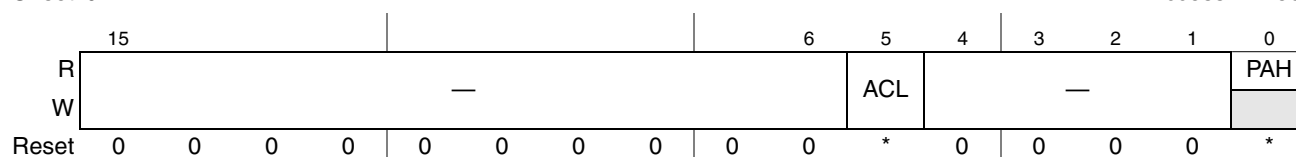
Bits	Name	Description
7-0	MAXLAT	Specifies how often the device needs to gain access to the PCI bus (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers.)

### 17.3.2.19 PCI Bus Function Register (PBFR)

The 2-byte PCI bus function register is used to determine how different features of the PCI interface in bus 0 are configured. This register is in PCI configuration space at offset 0x44.

Offset 0x44

Access: Mixed



\* = Depends on the state of the reset configuration signals at reset

Figure 17-46. PCI Bus Function Register

Table 17-45. PCI Bus Function Register Field Descriptions

Bits	Name	Description
15-6	—	Reserved
5	ACL	Agent configuration lock. Indicates to an external host whether the local processor is doing internal configuration and must be explicitly set and cleared by the local processor during this time. ACL is set during reset if the <code>cfg_cpu_boot</code> configuration input selects the CPU as the configuration owner. (See <a href="#">Section 4.4.3.7, “CPU Boot Configuration.”</a> ) This bit is only meaningful in agent mode. 0 PCI interface allows incoming PCI configuration cycles. 1 PCI interface retries all incoming PCI configuration cycles.
4-1	—	Reserved
0	PAH	PCI agent/host. Read-only. Indicates the reset value of the <code>cfg_host_agt</code> configuration signal. 0 PCI interface is in host mode 1 PCI interface is in agent mode

### 17.3.2.20 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

Offset 0x46

Access: Mixed

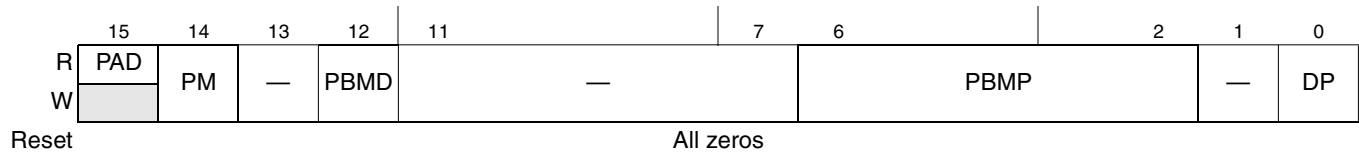


Figure 17-47. PCI Bus Arbiter Configuration Register

Table 17-46. PCI Bus Arbiter Configuration Register Field Descriptions

Bits	Name	Description
15	PAD	PCI arbiter disable. Determines if the device is the PCI arbiter on the PCI bus or not. The reset state is determined by the inverse of the <code>cfg_pcin_arb</code> configuration input signal when reset is released. 0 Device is the PCI arbiter. 1 Device is not the PCI arbiter. Device presents its request on $\overline{\text{PCI\_REQ0}}$ to the external arbiter and receives its grant on $\overline{\text{PCI\_GNT0}}$ .
14	PM	Parking mode. controls which device receives the bus grant when there are no outstanding bus requests and the bus is idle. 0 The bus is parked on the last device to use the bus. 1 The bus is parked on the device.
13	—	Reserved
12	PBMD	PCI broken master disable. Determines if the device ignores the bus requests of an initiator that requests the bus for an excessive period without using the bus. 0 An initiator that requests the bus and receives the grant must begin using the bus within 16 PCI clock periods after the bus becomes idle or else its request is subsequently ignored. 1 No requests are ignored.
11–7	—	Reserved
6–2	PBMP	PCI bus master priorities. Determines arbitration priority given to different masters on the PCI bus. Bit 6 corresponds to the priority of the master sourcing $\overline{\text{PCI\_REQ0}}$ ; bit 2 corresponds to the priority of the master sourcing $\overline{\text{PCI\_REQ4}}$ . 0 Master <i>n</i> is low priority. 1 Master <i>n</i> is high priority.
1	—	Reserved
0	DP	Device priority. Determines this device's arbitration priority. 0 Device is low priority. 1 Device is high priority.

## 17.4 Functional Description

This section describes the functionality of the PCI interface.

### 17.4.1 PCI Bus Arbitration

PCI bus arbitration is access-based. Bus masters must arbitrate for each access performed on the bus. The PCI bus uses a central arbitration scheme where each master has its own unique request ( $\overline{\text{REQ}}$ ) output and



grant ( $\overline{\text{GNT}}$ ) input signal. A simple request/grant handshake is used to gain access to the bus. Arbitration for the bus occurs during the previous access so that no PCI bus cycles are consumed due to arbitration (except when the bus is idle).

The PCI controller provides bus arbitration logic for its master interface and up to five other external PCI bus masters. The on-chip PCI arbiter is independent of host or agent mode. The on-chip PCI arbiter functions in both host and agent modes, or it can be disabled to allow for an external PCI arbiter.

A configuration signal sampled at the negation of the reset signal ( $\overline{\text{HRESET}}$ ) determines if the on-chip PCI arbiter is enabled (high) or disabled (low). The on-chip PCI arbiter can also be enabled or disabled by programming bit 15 of the PCI bus arbitration control register (PBACR[*PAD*]). Note that the sense of PBACR[*PAD*] corresponds to the inverse of the configuration signal (that is, when *PAD* = 0 the arbiter is enabled, and when *PA* = 1 the arbiter is disabled). See [Chapter 4, “Reset, Clocking, and Initialization,”](#) for more information on the reset configuration signals.

If the on-chip PCI arbiter is enabled, a request-grant pair of signals is provided for each external master ( $\overline{\text{PCI\_REQ}}[0:4]$  and  $\overline{\text{PCI\_GNT}}[0:4]$ ). In addition, there is an internal request/grant pair for the internal master state machine that governs internal accesses to the PCI interface. If the on-chip PCI arbiter is disabled, the PCI controller uses the  $\overline{\text{PCI\_REQ0}}$  signal as an output to issue its request to the external arbiter and uses the  $\overline{\text{PCI\_GNT0}}$  signal as an input to receive its grant from the external arbiter.

The following sections describe the operation of the on-chip PCI arbiter that arbitrates between external PCI masters and the internal PCI bus master.

### 17.4.1.1 PCI Bus Arbiter Operation

The on-chip PCI arbiter uses a programmable two-level, round-robin arbitration algorithm. Each of the five external masters, plus the device itself, can be programmed for two priority levels, high or low, using the appropriate bits in the PBACR. Within each priority group, the PCI bus grant is asserted to the next requesting device in numerical order, with the PCI controller positioned before device 0.

Conceptually, the lowest priority device is the master that is currently using the bus, and the highest priority device is the device that follows the current master in numerical order and group priority. This is considered to be a fair algorithm, since a single device cannot prevent other devices from having access to the bus; it automatically becomes the lowest priority device as soon as it begins to use the bus. If a master is not requesting the bus, then its transaction slot is given to the next requesting device within its priority group.

A grant is awarded to the highest priority requesting device as soon as the current master begins a transaction; however, the granted device must wait until the bus is relinquished by the current master before initiating a transaction.

The grant given to a particular device may be removed and awarded to another higher priority device, whenever the higher priority device asserts its request. If the bus is idle when a device requests the bus, then the arbiter withholds the grant for one clock cycle. The arbiter re-evaluates the priorities of all requesting devices and grants the bus to the highest priority device in the following clock cycle. This allows a turnaround clock when a higher priority device is using address stepping or when the bus is parked.

The low-priority group collectively has one bus transaction request slot in the high-priority group. For N high-priority devices and M low-priority devices, each high-priority device is guaranteed at least 1 of N+1 bus transactions and each low-priority device is guaranteed at least 1 of  $(N+1) \times M$  bus transactions, with one low-priority device receiving the grant in 1 of N+1 bus transactions. If all devices are programmed to the same priority level, or if the low-priority group has only one device, the algorithm defaults to give each device an equal number of bus grants in round-robin sequence.

For the example in Figure 17-48, assume that several devices are requesting the bus. If two masters are in the high-priority group and three are in the low-priority group, each high-priority master is guaranteed at least one out of three transaction slots and each low-priority master is guaranteed one out of nine transaction slots.

In Figure 17-48, the grant sequence (with all devices, except device 4 requesting the bus and device 3 being the current master) is 0, 2, MPC8533E, 0, 2, 1, 0, 2, 3, ..., and repeating. If device 2 is not requesting the bus, the grant sequence is 0, MPC8533E, 0, 1, 0, 3, ..., and repeating. If device 2 requests the bus when device 0 is conducting a transaction and the MPC8533E has the next grant, the MPC8533E has its grant removed and device 2 is awarded the grant since device 2 is higher priority than the MPC8533E when device 0 has the bus.

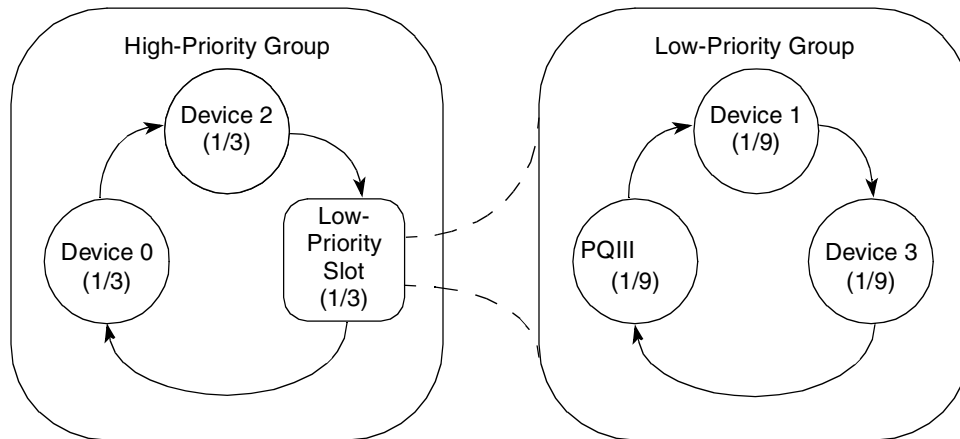


Figure 17-48. PCI Arbitration Example

### 17.4.1.2 PCI Bus Parking

When no device is using or requesting the bus, the PCI arbiter grants the bus to a selected device. This is known as parking the bus on the selected device. The selected device is required to drive the PCI\_AD[31:0], PCI\_C/ $\overline{BE}$ [0:3], and the PCI parity signals to a stable value, preventing these signals from floating.

The parking mode parameter (PBACR[PM]) determines which device the arbiter selects for parking the PCI bus. If PBACR[PM] = 0 (or if the bus is not idle), then the bus is parked on the last master to use the bus. If the bus is idle and PBACR[PM] = 1, the bus is parked on the PCI controller.

### 17.4.1.3 Broken Master Lock-Out

The PCI bus arbiter has a feature that allows it to lock out any masters that are broken or ill-behaved. The broken master feature is controlled by programming bit 12 of the PCI bus arbitration control register (0 = enabled, 1 = disabled).

When the broken master feature is enabled, a granted device that does not assert  $\overline{\text{PCI\_FRAME}}$  within 16 PCI clock cycles after the bus is idle, has its grant removed and subsequent requests are ignored until its  $\overline{\text{REQ}}$  is negated for at least one clock cycle. This prevents ill-behaved masters from monopolizing the bus. When the broken master feature is disabled, a device that requests the bus and receives a grant never loses its grant until and unless it begins a transaction or negates its REQ signal. Note that disabling the broken master feature is not recommended.

### 17.4.1.4 Power-Saving Modes and the PCI Arbiter

In the sleep power-saving mode, the clock signal driving SYSCLK can be disabled. If the clock is disabled, the arbitration logic is not able to perform its function. System programmers must park the bus with a device that can sustain the PCI\_AD[31:0], PCI\_C/ $\overline{\text{BE}}$ [3:0], and parity signals prior to disabling the SYSCLK signal. If the bus is parked on the MPC8533E when its clocks are stopped, the MPC8533E sustains the PCI\_AD[31:0], PCI\_C/ $\overline{\text{BE}}$ [3:0], and parity signals in their prior states. In this situation, the only way for another agent to use the PCI bus is by waking the MPC8533E. In nap and doze power-saving modes, the arbiter continues to operate allowing other PCI devices to run transactions.

## 17.4.2 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in [Section 17.4.2.7, “PCI Bus Transactions.”](#) Refer to [Figure 17-49](#), [Figure 17-50](#), [Figure 17-51](#), and [Figure 17-52](#) for examples of the transfer-control mechanisms described in this section.

All signals are sampled on the rising edge of the PCI bus clock (SYSCLK). Each signal has a setup and hold aperture with respect to the rising clock edge in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance. See the separate hardware specifications document for specific setup and hold times.

### 17.4.2.1 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst. A burst is composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals— $\overline{\text{PCI\_FRAME}}$  (frame),  $\overline{\text{PCI\_IRDY}}$  (initiator ready), and  $\overline{\text{PCI\_TRDY}}$  (target ready). An initiator asserts  $\overline{\text{PCI\_FRAME}}$  to indicate the beginning of a PCI bus transaction and negates  $\overline{\text{PCI\_FRAME}}$  to indicate the end of a PCI bus transaction. An initiator negates  $\overline{\text{PCI\_IRDY}}$  to force wait cycles. A target negates  $\overline{\text{PCI\_TRDY}}$  to force wait cycles.

The PCI bus is considered idle when both  $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  are negated. The first clock cycle in which  $\overline{\text{PCI\_FRAME}}$  is asserted indicates the beginning of the address phase. The address and bus command code are transferred in that first cycle. The next cycle begins the first of one or more data phases. Data is transferred between initiator and target in each cycle that both  $\overline{\text{PCI\_IRDY}}$  and  $\overline{\text{PCI\_TRDY}}$  are

asserted. Wait cycles may be inserted in a data phase by the initiator (by negating  $\overline{\text{PCI\_IRDY}}$ ) or by the target (by negating  $\overline{\text{PCI\_TRDY}}$ ).

Once an initiator has asserted  $\overline{\text{PCI\_IRDY}}$ , it cannot change  $\overline{\text{PCI\_IRDY}}$  or  $\overline{\text{PCI\_FRAME}}$  until the current data phase completes regardless of the state of  $\overline{\text{PCI\_TRDY}}$ . Once a target has asserted  $\overline{\text{PCI\_TRDY}}$  or  $\overline{\text{PCI\_STOP}}$ , it cannot change  $\overline{\text{PCI\_DEVSEL}}$ ,  $\overline{\text{PCI\_TRDY}}$ , or  $\overline{\text{PCI\_STOP}}$  until the current data phase completes. In simpler terms, once an initiator or target has committed to the data transfer, it cannot change its mind.

When the initiator intends to complete only one more data transfer (which could be immediately after the address phase),  $\overline{\text{PCI\_FRAME}}$  is negated and  $\overline{\text{PCI\_IRDY}}$  is asserted (or kept asserted), indicating the initiator is ready. After the target indicates the final data transfer (by asserting  $\overline{\text{PCI\_TRDY}}$ ), the PCI bus may return to the idle state (both  $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  are negated) unless a fast back-to-back transaction is in progress. In the case of a fast back-to-back transaction, an address phase immediately follows the last data phase.

### 17.4.2.2 PCI Bus Commands

A PCI bus command is encoded in the  $\text{PCI\_C}/\overline{\text{BE}}[3:0]$  signals during the address phase of a PCI transaction. The bus command indicates to the target the type of transaction the initiator is requesting. [Table 17-47](#) describes the PCI bus commands implemented by the device.

**Table 17-47. PCI Bus Commands**

PCI_C/ $\overline{\text{BE}}[3:0]$	PCI Bus Command	Supported as an Initiator	Supported as a Target	Definition
0000	Interrupt-acknowledge	Yes	No	A read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to this command; others ignore it. See <a href="#">Section 17.4.2.12.1, “Interrupt-Acknowledge Transactions,”</a> for more information.
0001	Special cycle	Yes	No	Provides a way to broadcast select messages to all devices on the PCI bus. See <a href="#">Section 17.4.2.12.2, “Special-Cycle Transactions,”</a> for more information.
0010	I/O-read	Yes	No	Accesses agents mapped into the PCI I/O space.
0011	I/O-write	Yes	No	Accesses agents mapped into the PCI I/O space.
0100	Reserved <sup>1</sup>	No	No	—
0101	Reserved <sup>1</sup>	No	No	—
0110	Memory-read	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address. When a PCI master issues this command to local memory, the PCI controller (the target) fetches data from the requested address to the end of the cache line (32 bytes) from local memory, even though all of the data may not be requested by (or sent to) the initiator.
0111	Memory-write	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address.
1000	Reserved <sup>1</sup>	No	No	—
1001	Reserved <sup>1</sup>	No	No	—

Table 17-47. PCI Bus Commands (continued)

PCI_C/ BE[3:0]	PCI Bus Command	Supported as an Initiator	Supported as a Target	Definition
1010	Configuration-read	Yes	Agent mode only	Accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See <a href="#">Section 17.4.2.11, “Configuration Cycles,”</a> for details.
1011	Configuration-write	Yes	Agent mode only	
1100	Memory-read-multiple	Yes	Yes	Similar to the memory-read command, but also causes a prefetch of the next cache line (32 bytes).
1101	Dual-address-cycle	Yes	Yes	Used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices.
1110	Memory-read-line	Yes	Yes	Indicates that an initiator is requesting the transfer of an entire cache line. This occurs only when the processor is performing a burst read. Note that these processors perform burst reads only when the appropriate cache is enabled and the transaction is not cache-inhibited.
1111	Memory-write-and-invalidate	No	Yes	Indicates that an initiator is transferring an entire cache line; if this data is in any cacheable memory, that cache line needs to be invalidated.

<sup>1</sup> Reserved command encodings are reserved for future use. The PCI controller does not respond to these commands.

### 17.4.2.3 Addressing

PCI defines three physical address spaces—PCI memory space, PCI I/O space, and PCI configuration space. Access to the PCI memory and I/O space is straightforward, although one must take into account the local memory access window and address translation being used. The address translation registers are described in [Section 17.3.1, “PCI Memory-Mapped Registers.”](#) Access to the PCI configuration space is described in [Section 17.4.2.11, “Configuration Cycles.”](#)

Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding—positive decoding and subtractive decoding. For positive decoding, each device looks for accesses in the address range that the device has been assigned. For subtractive decoding, one device on the bus looks for accesses that no other device has claimed. See [Section 17.4.2.4, “Device Selection,”](#) for information about claiming transactions.

The information contained in the two low-order address bits (PCI\_AD[1:0]) varies by the address space (memory, I/O, or configuration). Regardless of the encoding scheme, the two low-order address bits are always included in parity calculations.

#### 17.4.2.3.1 Memory Space Addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address—linear incrementing (PCI\_AD[1:0] = 0b00) and cache wrap mode (PCI\_AD[1:0] = 0b10), as shown in [Table 17-48](#). The other two PCI\_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the PCI controller always encodes PCI\_AD[1:0] = 00 for PCI memory space accesses. As a target, the PCI controller executes a target disconnect after the first data phase completes if

PCI\_AD[1:0] = 01 or PCI\_AD[1:0] = 0b11 during the address phase of a local memory access. See [Section 17.4.2.8.2, “Target-Initiated Termination,”](#) for more information on target disconnect conditions.

**Table 17-48. Supported Combinations of PCI\_AD[1:0]**

PCI_AD[1:0]		Target		Initiator	
		Read	Write	Read	Write
00	Linear	√	√	√	√
01	Reserved	TD	TD	—	—
10	Cache Wrap	√	TD	—	—
11	Reserved	TD	TD	—	—

For linear incrementing mode, the memory address is encoded/decoded using PCI\_AD[31:2]. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4-byte data width per data phase is implied). Note that the two low-order bits on the address bus are included in all parity calculations.

For cache wrap mode (PCI\_AD[1:0] = 0b10) reads, the critical memory address is decoded using PCI\_AD[31:2]. The address is incremented by 4 bytes after each data phase completes until the end of the cache line is reached. For cache-wrap reads, the address wraps to the beginning of the current cache line and continues incrementing until the entire cache line (32 bytes) is read. The PCI controller does not support cache-wrap write operations and executes a target disconnect after the data phase for the end of the cache line completes for writes with PCI\_AD[1:0] = 0b10. That is, the PCI controller does not wrap back to the beginning of the cache line. Note that the two low-order bits on the address bus are included in all parity calculations.

### 17.4.2.3.2 I/O Space Addressing

For PCI I/O accesses, 32 address signals (PCI\_AD[31:0]) are used to provide a byte address. After a target has claimed an I/O access, it must determine if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range of the target, the entire access cannot complete. In this case, the target does not transfer any data and terminates the transaction with a target-abort error. See [Section 17.4.2.8.2, “Target-Initiated Termination,”](#) for more information.

### 17.4.2.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses that use different formats for the PCI\_AD[31:0] signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase—type 0 (PCI\_AD[1:0] = 0b00) or type 1 (PCI\_AD[1:0] = 0b01). Both address formats identify a specific device and a specific configuration register for that device. See [Section 17.4.2.11, “Configuration Cycles,”](#) for descriptions of the two formats.

### 17.4.2.4 Device Selection

The  $\overline{\text{PCI\_DEVSEL}}$  signal is driven by the target of the current transaction.  $\overline{\text{PCI\_DEVSEL}}$  indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction.



$\overline{\text{PCI\_DEVSEL}}$  may be driven one, two, or three clock cycles (fast, medium, or slow device select timing) following the address phase. Device select timing is encoded into the device's PCI bus status register. If no agent asserts  $\overline{\text{PCI\_DEVSEL}}$  within three clock cycles of  $\overline{\text{PCI\_FRAME}}$ , the agent responsible for subtractive decoding may claim the transaction by asserting  $\overline{\text{PCI\_DEVSEL}}$ .

A target must assert  $\overline{\text{PCI\_DEVSEL}}$  (claim the transaction) before or coincident with any other target response (assert  $\overline{\text{PCI\_TRDY}}$ ,  $\overline{\text{PCI\_STOP}}$ , or data signals). In all cases except target-abort, once a target asserts  $\overline{\text{PCI\_DEVSEL}}$ , it must not negate  $\overline{\text{PCI\_DEVSEL}}$  until  $\overline{\text{PCI\_FRAME}}$  is negated (with  $\overline{\text{PCI\_IRDY}}$  asserted) and the last data phase has completed. For normal termination, negation of  $\overline{\text{PCI\_DEVSEL}}$  coincides with the negation of  $\overline{\text{PCI\_TRDY}}$  or  $\overline{\text{PCI\_STOP}}$ .

If the first access maps into a target's address range, that target asserts  $\overline{\text{PCI\_DEVSEL}}$  to claim the access. However, if the initiator attempts to continue the burst access across the resource boundary, then the target must issue a target disconnect.

The PCI controller is hardwired for fast device select timing (PCI bus status register [10–9] = 0b00). Therefore, when the PCI controller is the target of a transaction (local memory access or configuration register access), it asserts  $\overline{\text{PCI\_DEVSEL}}$  one clock cycle following the address phase.

As an initiator, if the PCI controller does not detect the assertion of  $\overline{\text{PCI\_DEVSEL}}$  within four clock cycles after the address phase (that is, five clock cycles after it asserts  $\overline{\text{PCI\_FRAME}}$ ), it terminates the transaction with a master-abort termination; see [Section 17.4.2.8.1, "Master-Initiated Termination."](#)

#### 17.4.2.5 Byte Alignment

The byte enable signals of the PCI bus ( $\overline{\text{PCI\_C/BE}}[3:0]$ , during a data phase) are used to determine which byte lanes carry meaningful data. The byte enable signals may enable different bytes for each of the data phases. The byte enables are valid on the edge of the clock that starts each data phase and stay valid for the entire data phase. Note that parity is calculated for all bytes regardless of the state of the byte enable signals. See [Section 17.4.2.13.1, "PCI Parity,"](#) for more information.

If the PCI controller, as a target, detects no byte enables asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the PCI controller expects that the data is not changed, and on a write transaction, the data is not stored.

#### 17.4.2.6 Bus Driving and Turnaround


To avoid contention, a turnaround cycle is required on all signals that may be driven by more than one agent. The turnaround cycle occurs at different times for different signals. The  $\overline{\text{PCI\_IRDY}}$ ,  $\overline{\text{PCI\_TRDY}}$ ,  $\overline{\text{PCI\_DEVSEL}}$ , and  $\overline{\text{PCI\_STOP}}$  signals use the address phase as their turnaround cycle.  $\overline{\text{PCI\_FRAME}}$ ,  $\overline{\text{PCI\_C/BE}}[3:0]$ , and  $\overline{\text{PCI\_AD}}[31:0]$  signals use the idle cycle between transactions (when both  $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  are negated) as their turnaround cycle.  $\overline{\text{PCI\_PERR}}$  has a turnaround cycle on the fourth clock cycle after the last data phase.

The PCI address/data signals,  $\overline{\text{PCI\_AD}}[31:0]$ , are driven to a stable condition during every address/data phase. Even when the byte enables indicate that byte lanes carry meaningless data, the signals carry stable values. Parity is calculated on all bytes regardless of the byte enables. See [Section 17.4.2.13.1, "PCI Parity,"](#) for more information.

### 17.4.2.7 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in [Section 17.4.2, “PCI Bus Protocol.”](#) Read and write transactions are similar for the memory and I/O spaces, so they are described as generic read transactions and generic write transactions.

The timing diagrams in this section show the relationship of significant signals involved in bus transactions. When a signal is drawn as a solid line, it is actively being driven by the current master or target. When a signal is drawn as a dashed line, no agent is actively driving it. High-impedance signals are indicated to have indeterminate values when the dashed line is between the two rails.

The terms ‘edge’ and ‘clock edge’ always refer to the rising edge of the clock. The terms ‘asserted’ and ‘negated’ always refer to the globally visible state of the signal on the clock edge, and not to signal transitions. ‘’ represents a turnaround cycle in the timing diagrams.

#### 17.4.2.7.1 PCI Read Transactions

This section describes PCI single-beat read transactions and PCI burst read transactions.

A read transaction starts with the address phase, occurring when an initiator asserts  $\overline{\text{PCI\_FRAME}}$ . During the address phase,  $\text{PCI\_AD}[31:0]$  contains a valid address and  $\text{PCI\_C}/\overline{\text{BE}}[3:0]$  contains a valid bus command.

The first data phase of a read transaction requires a turnaround cycle. This allows the transition from the initiator driving  $\text{PCI\_AD}[31:0]$  as address signals to the target driving  $\text{PCI\_AD}[31:0]$  as data signals. The turnaround cycle is enforced by the target with the  $\overline{\text{TRDY}}$  signal. The target provides valid data at the earliest one cycle after the turnaround cycle. The target must drive the  $\text{PCI\_AD}[31:0]$  signals when  $\text{PCI\_DEVSEL}$  is asserted.

During the data phase, the  $\text{PCI\_C}/\overline{\text{BE}}[3:0]$  signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The  $\text{PCI\_C}/\overline{\text{BE}}[3:0]$  signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both  $\overline{\text{PCI\_IRDY}}$  and  $\overline{\text{PCI\_TRDY}}$  are asserted on the same clock edge. When either  $\overline{\text{PCI\_IRDY}}$  or  $\overline{\text{PCI\_TRDY}}$  is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating  $\overline{\text{PCI\_FRAME}}$  when  $\overline{\text{PCI\_IRDY}}$  is asserted. The transaction is considered complete when data is transferred in the last data phase.



Figure 17-49 illustrates a PCI single-beat read transaction.

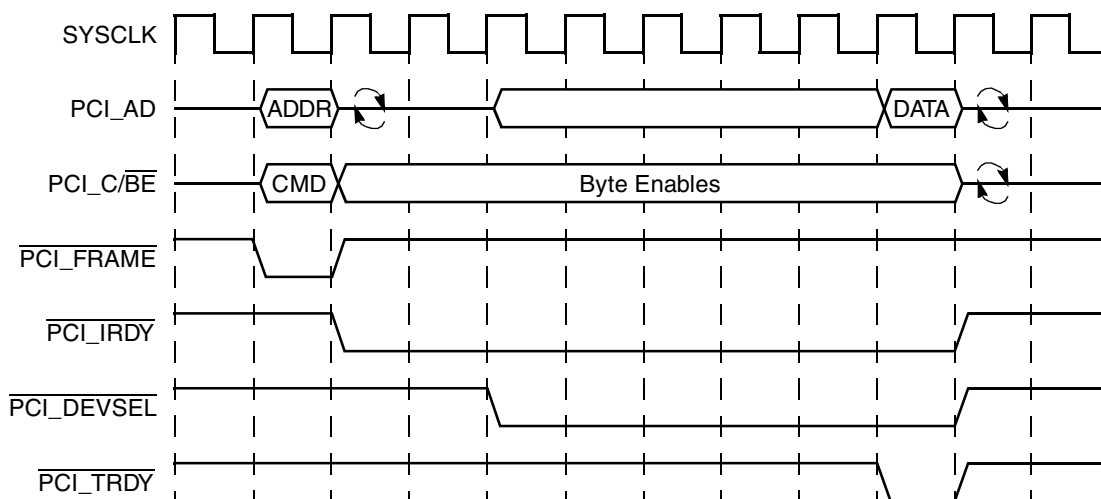


Figure 17-49. PCI Single-Beat Read Transaction

Figure 17-50 illustrates a PCI burst read transaction.

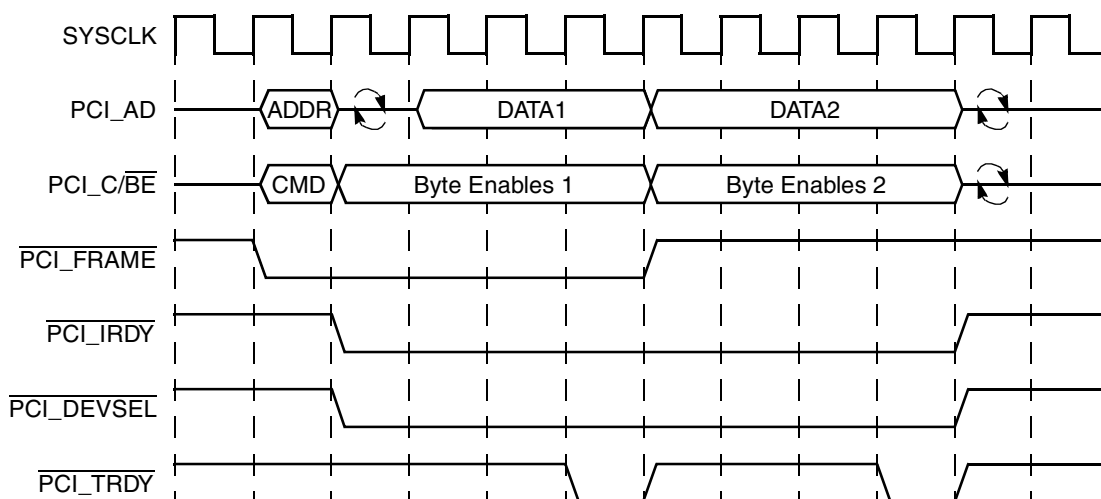


Figure 17-50. PCI Burst Read Transaction

#### 17.4.2.7.2 PCI Write Transactions

This section describes PCI single-beat write transactions, and PCI burst write transactions. A PCI write transaction starts with the address phase, occurring when an initiator asserts  $\overline{\text{PCI\_FRAME}}$ . A write transaction is similar to a read transaction except no turnaround cycle is needed following the address phase because the initiator provides both address and data. The data phases are the same for both read and write transactions. Although not shown in the figures, the initiator must drive the  $\text{PCI\_C/BE}[3:0]$  signals, even if the initiator is not ready to provide valid data ( $\overline{\text{PCI\_IRDY}}$  negated).

Figure 17-51 illustrates a PCI single-beat write transaction.

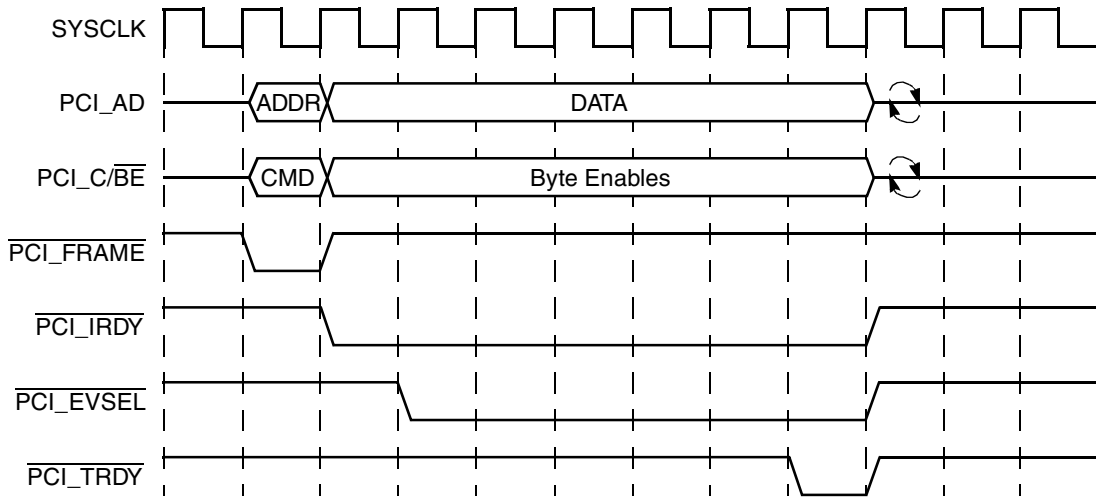


Figure 17-51. PCI Single-Beat Write Transaction

Figure 17-52 illustrates a PCI burst write transaction.

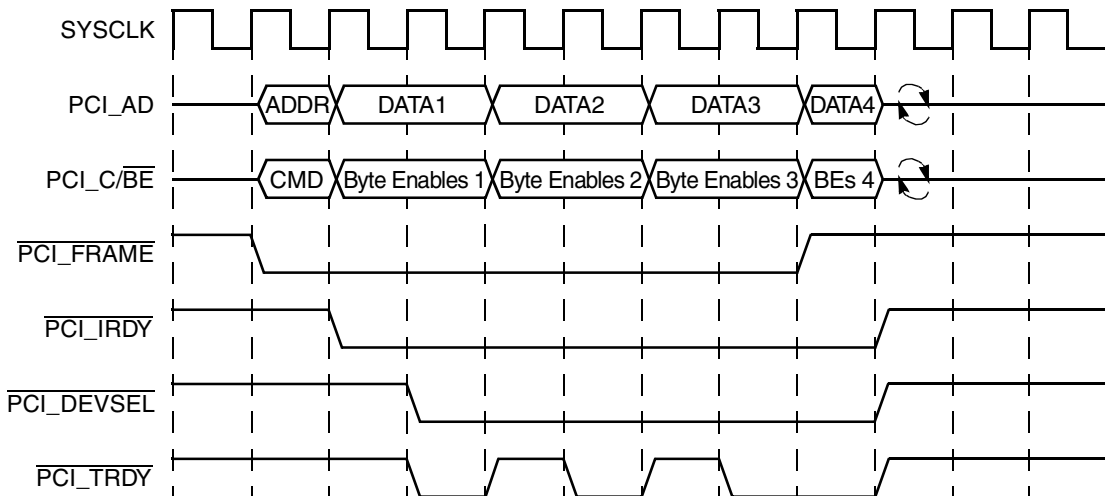


Figure 17-52. PCI Burst Write Transaction

### 17.4.2.8 Transaction Termination

A PCI transaction may be terminated by either the initiator or the target. The initiator is ultimately responsible for concluding all transactions, regardless of the cause of the termination. All transactions are concluded when  $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  are both negated, indicating the bus is idle.

#### 17.4.2.8.1 Master-Initiated Termination

Normally, a master initiates termination by negating  $\overline{\text{PCI\_FRAME}}$  and asserting  $\overline{\text{PCI\_IRDY}}$ . This indicates to the target that the final data phase is in progress. The final data transfer occurs when both  $\overline{\text{PCI\_TRDY}}$  and  $\overline{\text{PCI\_IRDY}}$  are asserted. The transaction is considered complete when data is transferred

in the last data phase. After the final data phase, both  $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  are negated (the bus becomes idle).

There are three types of master-initiated termination:

- Completion—Refers to termination when the initiator has concluded its intended transaction. This is the most common reason for termination.
- Timeout—Refers to termination when the initiator loses its bus grant ( $\overline{\text{GNTn}}$  is negated), and its internal latency timer has expired. The intended transaction is not necessarily concluded.
- Master-abort—An abnormal case of master-initiated termination. If no device (including the subtractive decoding agent) asserts  $\overline{\text{PCI\_DEVSEL}}$  to claim a transaction, the initiator terminates the transaction with a master-abort. For a master-abort termination, the initiator negates  $\overline{\text{PCI\_FRAME}}$  and then negates  $\overline{\text{PCI\_IRDY}}$  on the next clock. If a transaction is terminated by master-abort (except for a special-cycle command), the received master-abort bit (bit 13) of the PCI bus status register is set.

As an initiator, if the PCI controller does not detect the assertion of  $\overline{\text{PCI\_DEVSEL}}$  within four clock cycles following the address phase (five clock cycles after asserting  $\overline{\text{PCI\_FRAME}}$ ), it terminates the transaction with a master-abort.

#### 17.4.2.8.2 Target-Initiated Termination

By asserting the  $\overline{\text{PCI\_STOP}}$  signal, a target may request that the initiator terminate the current transaction. Once asserted, the target holds  $\overline{\text{PCI\_STOP}}$  asserted until the initiator negates  $\overline{\text{PCI\_FRAME}}$ . Data may or may not be transferred during the request for termination. If  $\overline{\text{PCI\_TRDY}}$  and  $\overline{\text{PCI\_IRDY}}$  are asserted during the assertion of  $\overline{\text{PCI\_STOP}}$ , data is transferred. However, if  $\overline{\text{PCI\_TRDY}}$  is negated when  $\overline{\text{PCI\_STOP}}$  is asserted, it indicates that the target will not transfer any more data; therefore, the initiator does not wait for a final data transfer as it would in a completion termination.

When a transaction is terminated by  $\overline{\text{PCI\_STOP}}$ , the initiator must negate its  $\overline{\text{REQn}}$  signal for a minimum of two PCI clock cycles, (one corresponding to when the bus goes to the idle state ( $\overline{\text{PCI\_FRAME}}$  and  $\overline{\text{PCI\_IRDY}}$  negated)). If the initiator intends to complete the transaction, it can reassert its  $\overline{\text{REQn}}$  immediately following the two clock cycles. If the initiator does not intend to complete the transaction, it can assert  $\overline{\text{REQn}}$  whenever it needs to use the PCI bus again.

There are three types of target-initiated termination:

- Disconnect—Disconnect refers to termination requested because the target is temporarily unable to continue bursting. Disconnect implies that some data has been transferred. The initiator may restart the transaction at a later time starting with the address of the next untransferred data. (That is, data transfer may resume where it left off.)
- Retry—Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. Retry implies that no data was transferred. The initiator may start the entire transaction over again at a later time. Note that the *PCI Local Bus Specification, Rev. 2.2* requires that all retried transactions must be completed.
- Target-Abort—Target-abort is an abnormal case of target-initiated termination. Target-abort is used when a fatal error has occurred or when a target can never respond.

As a target, the PCI controller terminates a transaction with a target disconnect due to the following:

- It is unable to respond within eight PCI clock cycles (not including the first data phase).
- The transaction is attempting to cross a 4-Kbyte boundary.
- A single beat of data has been transferred and the inbound ATMU is marked non-prefetchable.
- The end of a cache line has been transferred for a cache-wrap mode write transaction. See [Section 17.4.2.3.1, “Memory Space Addressing,”](#) for more information.

As a target, the PCI controller responds to a transaction with a retry due to the following:

- The 32-clock latency timer has expired, and the first data phase has not begun.
- There is no more internal buffer space available for an inbound transaction.

Target-abort is indicated by asserting  $\overline{\text{PCI\_STOP}}$  and negating  $\overline{\text{PCI\_DEVSEL}}$ . This indicates that the target requires termination of the transaction and does not want the transaction retried. If a transaction is terminated by target-abort, the received target-abort bit (bit 12) of the initiator’s bus status register and the signaled target-abort bit (bit 11) of the target’s bus status register are set. Note that any data transferred in a target-aborted transaction may be corrupt.

For PCI writes to local memory, if an address parity error or data parity error occurs, the PCI controller aborts the transaction internally, but continues the transaction on the PCI bus.

Figure 17-53 shows several target-initiated terminations.

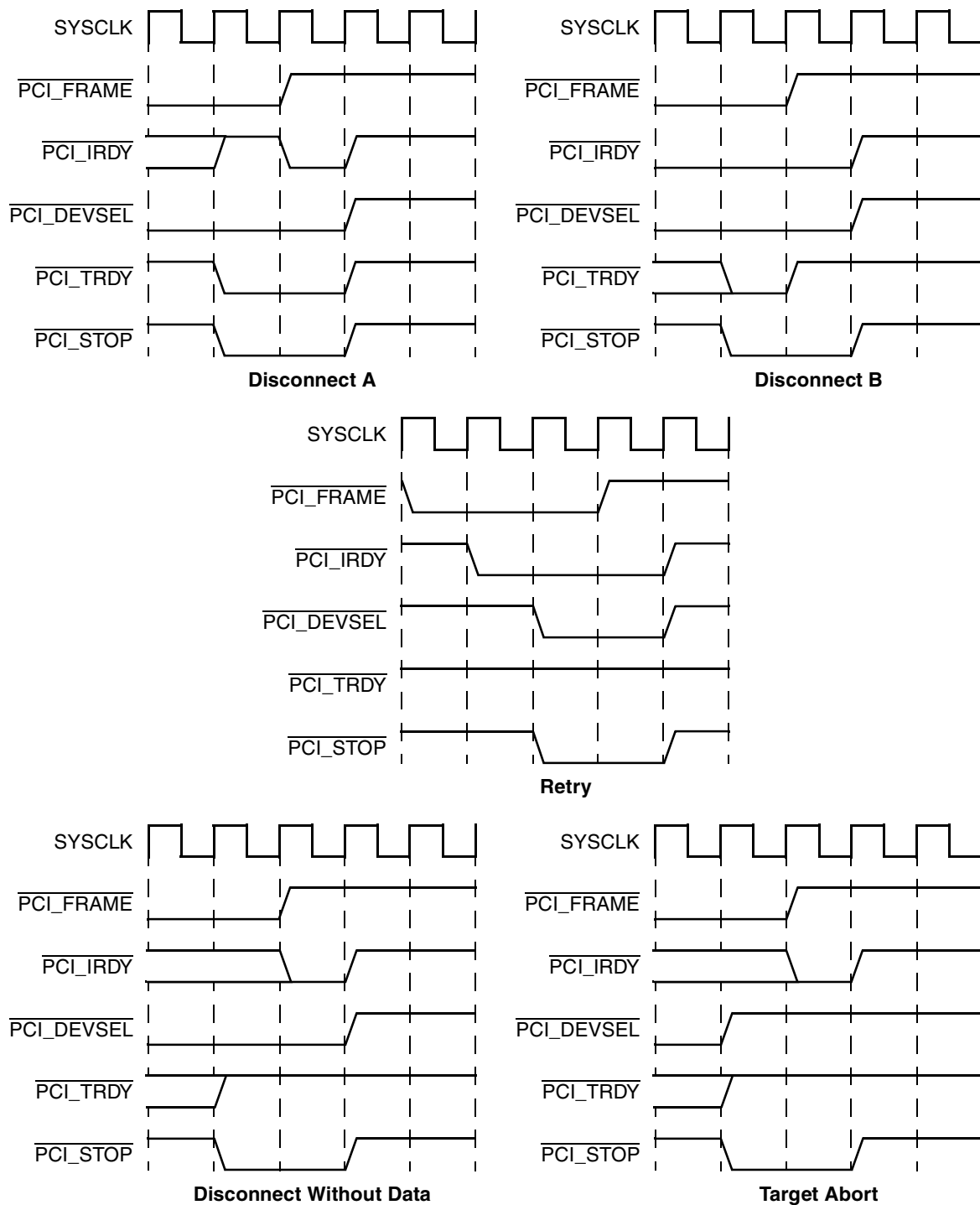


Figure 17-53. PCI Target-Initiated Terminations

The three disconnect terminations are unique in the data transferred at the end of the transaction. For disconnect A, the initiator is negating  $\overline{\text{PCI\_IRDY}}$  when the target asserts  $\overline{\text{PCI\_STOP}}$  and data is transferred only at the end of the current data phase. For disconnect B, the target negates  $\overline{\text{PCI\_TRDY}}$  one clock after

it asserts  $\overline{\text{PCI\_STOP}}$ , indicating that the target can accept the current data, but no more data can be transferred. For disconnect-without-data, the target asserts  $\overline{\text{PCI\_STOP}}$  when  $\overline{\text{PCI\_TRDY}}$  is negated indicating that the target cannot accept any more data.

### 17.4.2.9 Fast Back-to-Back Transactions

The PCI bus allows fast back-to-back transactions by the same master. During a fast back-to-back transaction, the initiator starts the next transaction immediately without an idle state. The last data phase completes when  $\overline{\text{PCI\_FRAME}}$  is negated, and  $\overline{\text{PCI\_IRDY}}$  and  $\overline{\text{PCI\_TRDY}}$  are asserted. The current master starts another transaction in the clock cycle immediately following the last data transfer for the previous transaction.

Fast back-to-back transactions must avoid contention on the  $\overline{\text{PCI\_TRDY}}$ ,  $\overline{\text{PCI\_DEVSEL}}$ ,  $\overline{\text{PCI\_PERR}}$ , and  $\overline{\text{PCI\_STOP}}$  signals. There are two types of fast back-to-back transactions—those that access the same target and those that access multiple targets sequentially. The first type places the burden of avoiding contention on the initiator; the second type places the burden of avoiding contention on all potential targets.

As an initiator, the PCI controller does not perform any fast back-to-back transactions. As a target, the PCI controller supports both types of fast back-to-back transactions.

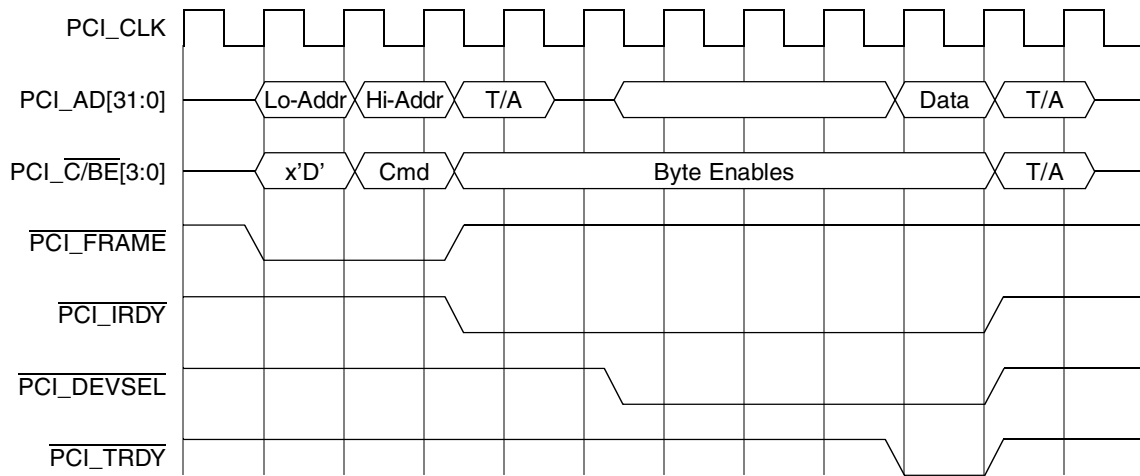
During fast back-to-back transactions, the PCI controller monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the PCI controller but the current transaction is directed at the PCI controller, it delays the assertion of  $\overline{\text{PCI\_DEVSEL}}$  (as well as  $\overline{\text{PCI\_TRDY}}$ ,  $\overline{\text{PCI\_STOP}}$ , and  $\overline{\text{PCI\_PERR}}$ ) for one clock cycle to allow the other target to stop driving the bus.

### 17.4.2.10 Dual Address Cycles

The PCI controller supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as both an initiator and a target. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The PCI controller block supports single-beat and burst DAC transactions.

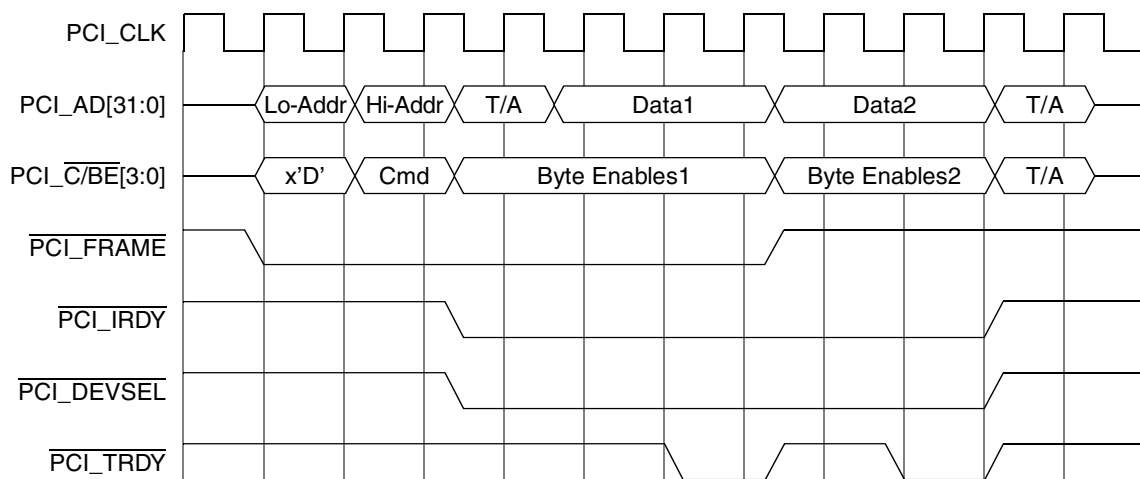
For the case of the local processor, DAC generation depends on the setting of the POTEARx. If the POTEARx are programmed with nonzero values and a transaction from the local processor core hits in one of the outbound windows, a DAC transaction is generated on the PCI bus with the translated lower 32-bit addresses. Refer to [Section 17.3.1.2, “PCI ATMU Outbound Registers,”](#) for more information.

The timing sequence of the PCI signals for single-beat DAC reads is shown in [Figure 17-54](#).



**Figure 17-54. DAC Single-Beat Read Example**

The timing for a DAC burst read is shown in [Figure 17-55](#).



**Figure 17-55. DAC Burst Read Example**

Figure 17-56 and Figure 17-57 show timing examples for single-beat DAC writes and burst DAC writes, respectively.

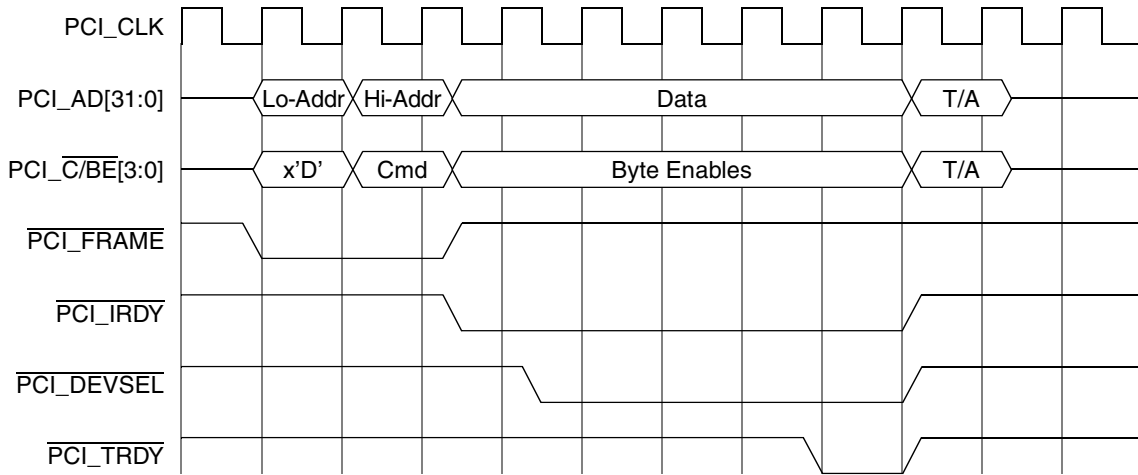


Figure 17-56. DAC Single-Beat Write Example

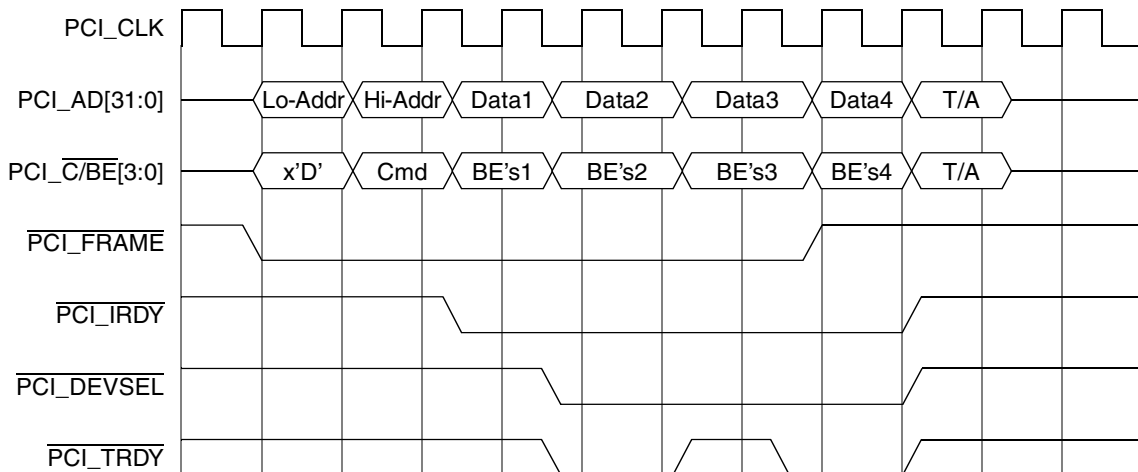


Figure 17-57. DAC Burst Write Example

### 17.4.2.11 Configuration Cycles

This section describes PCI configuration cycles used for configuring standard PCI devices. The PCI configuration space of any device is intended for configuration, initialization, and catastrophic error-handling functions only. Access to the PCI configuration space should be limited to initialization and error-handling software.

#### 17.4.2.11.1 PCI Configuration Space Header

The first 64 bytes of the 256-byte configuration space consists of a predefined header that every PCI device must support. The predefined header for all PCI devices is shown in Figure 17-58. The first 16 bytes of the predefined header are defined the same for all PCI devices; the remaining 48 bytes of the header may have differing layouts depending on the function of the device. Most PCI devices use the configuration header



layout shown in [Figure 17-58](#). The rest of the 256-byte configuration space is device-specific. The PCI header specific to the PCI controller is described in [Section 17.3.2, “PCI Configuration Header.”](#)

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Registers				10
				14
				18
				1C
				20
Reserved				24
Reserved				28
Subsystem ID		Subsystem Vendor ID		2C
Expansion ROM Base Address				30
Reserved				34
Reserved				38
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3C

**Figure 17-58. Standard PCI Configuration Header**

[Table 17-49](#) summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification, Rev. 2.2*.

**Table 17-49. PCI Configuration Space Header Summary**

Address Offset (Hex)	Register Name	Description
0x00	Vendor ID	Identifies the manufacturer of the device (assigned by the PCI SIG (special-interest group) to ensure uniqueness).
0x02	Device ID	Identifies the particular device (assigned by the vendor).
0x04	Command	Provides coarse control over a device's ability to generate and respond to PCI bus cycles
0x06	Status	Records status information for PCI bus-related events
0x08	Revision ID	Specifies a device-specific revision code (assigned by vendor)
0x09	Class code	Identifies the generic function of the device and (in some cases) a specific register-level programming interface
0x0C	Cache line size	Specifies the system cache line size in 32-bit units
0x0D	Latency timer	Specifies the value of the latency timer in PCI bus clock units for the device when acting as an initiator
0x0E	Header type	Bits 0–6 identify the layout of bytes 0x10–0x3F; bit 7 indicates a multifunction device. The most common header type (0x00) is shown in <a href="#">Figure 17-58</a> and in this table.
0x0F	BIST	Optional register for control and status of built-in self test (BIST)
0x10–0x27	Base address registers	Address mapping information for memory and I/O space

Table 17-49. PCI Configuration Space Header Summary (continued)

Address Offset (Hex)	Register Name	Description
0x28	—	Reserved for future use
0x2C	Subsystem Vendor ID	Identifies the subsystem vendor ID
0x2E	Subsystem ID	Identifies the subsystem ID
0x30	Expansion ROM base address	Base address and size information for expansion ROM contained in an add-on board
0x34, 0x38	—	Reserved for future use
0x3C	Interrupt line	Contains interrupt line routing information
0x3D	Interrupt pin	Indicates which interrupt pin the device (or function) uses
0x3E	Min_Gnt	Specifies the length of the device's burst period in 0.25 $\mu$ s units
0x3F	Max_Lat	Specifies how often the device needs access to the bus in 0.25 $\mu$ s units

To access the configuration space, a 32-bit value must be written to the PCI CFG\_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. A read or write to the PCI CFG\_DATA register causes the host bridge to translate the access into a PCI configuration cycle (provided the enable bit in CONFIG\_ADDR is set and the device number is not 0b1\_1111).

See [Section 17.3.1.1.1, “PCI Configuration Address Register \(CFG\\_ADDR\),”](#) for details on PCI CFG\_ADDR and [Section 17.3.1.1.2, “PCI Configuration Data Register \(CFG\\_DATA\),”](#) for details on PCI CFG\_DATA.

#### 17.4.2.11.2 Host Accessing the PCI Configuration Space

Power Architecture processor accesses to the PCI CFG\_DATA register should use the load/store with byte-reversed instructions.

**Example:** Configuration sequence, 4-byte data read from the revision ID/standard programming interface/subclass code/class code registers at address offset 0x08 of the PCI configuration header (device 0 on the PCI bus 0 is the PCI controller itself).

Initial values:

```
r0 contains 0x8000_0008
r1 contains CCSRBAR + BlockBase + 0x000 (Address of PCI CFG_ADDR register)
r2 contains CCSRBAR + BlockBase + 0x004 (Address of PCI CFG_DATA register)
r3 contains 0xFFFF_FFFF
Register at 0x08 contains 0x9988_7766 (0x0B to 0x08)
```

Code sequence:

```
stw r0, 0 (r1)
ld r3, 0 (r2)
```

Results:

```
Address CCSRBAR + BlockBase + 0x000 contains 0x8000_0008
Register r3 contains 0x6677_8899
```

### 17.4.2.11.3 Agent Accessing the PCI Configuration Space

When this device is configured as an agent device, it responds to a remote host-generated PCI configuration cycle. This is indicated by decoding the configuration command along with PCI's IDSEL being asserted. When the PCI controller detects an access to PCI CFG\_DATA, it checks the enable flag and the device number in the PCI CFG\_ADDR register. If the enable bit is set, and the device number is not 0b1\_1111, the PCI controller performs a configuration cycle translation function and runs a configuration-read or configuration-write transaction on the PCI bus. The device number 0b1\_1111 is used for performing interrupt-acknowledge and special-cycle transactions. See [Section 17.4.2.12, “Other Bus Transactions,”](#) for more information. If the bus number corresponds to the local PCI bus (bus number = 0x00), the PCI controller performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the PCI controller performs a type 1 configuration cycle translation.

Note that in the following examples, the data in the configuration register is shown in little-endian order. This is because all the PCI registers are intrinsically little-endian. External PCI masters that use the local address map to access configuration space do not need to reverse bytes since byte lane redirection from the little-endian PCI bus is performed internally.

**Example:** Configuration sequence, 4-byte data write to PCI register at address offset 0x14 of Device 1 on PCI bus 0.

Initial values:

```
r0 contains 0x8000_0814
r1 contains CCSRBAR + BlockBase + 0x000 (Address of PCI CFG_ADDR register)
r2 contains CCSRBAR + BlockBase + 0x004 (Address of PCI CFG_DATA register)
r3 contains 0x1122_3344
Register at 0x14 contains 0xFFFF_FFFF (0x17 to 0x14)
```

Code sequence:

```
stw r0, 0 (r1) // Update PCI CFG_ADDR register to point to
                //register offset 0x14 of device 1.
stwbrx r3, 0 (r2)
```

Results:

```
Address CCSRBAR + BlockBase + 0x000 contains 0x8000_0814
Register at 0x14 contains 0x1122_3344 (0x17 to 0x14)
```

**Example:** Configuration sequence, 2-byte data write to PCI register at address offset 0x1C of Device 1 on PCI bus 0.

Initial values:

```
r0 contains 0x8000_081C
r1 contains CCSRBAR + BlockBase + 0x000
r2 contains CCSRBAR + BlockBase + 0x004
r3 contains 0xDDCC_BBAA
Register at 0x1C contains 0xFFFF_FFFF (0x1F to 0x1C)
```

Code sequence:

```
stw r0, 0 (r1)
sthbrx r3, 0 (r2)
```

Results:

```
Address CCSRBAR + BlockBase + 0x000 contains 0x8000_081C
Register at 0x1C contains 0xFFFF_BBAA (0x1F to 0x1C)
```

### 17.4.2.11.4 PCI Type 0 Configuration Translation

Figure 17-59 shows the PCI type 0 translation function performed on the contents of the PCI CFG\_ADDR register to the PCI\_AD[31:0] signals on the PCI bus during the address phase of the configuration cycle.

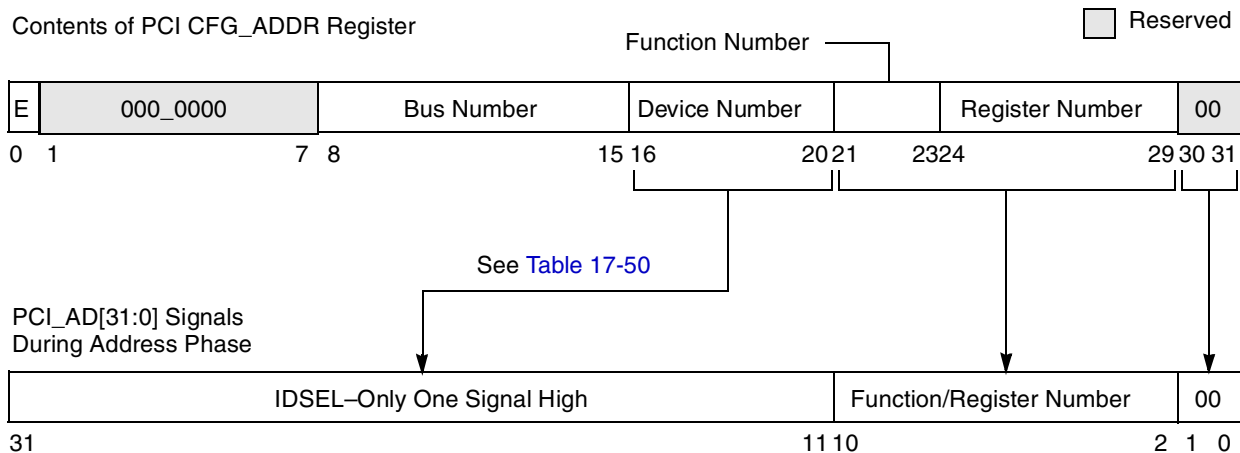


Figure 17-59. PCI Type 0 Configuration Translation

For PCI type 0 configuration cycles, the PCI controller translates the device number field of the PCI CFG\_ADDR register into a unique IDSEL signal for up to 21 different devices. Each device connects its IDSEL input to one of the PCI\_AD[31:11] signals. For PCI type 0 configuration cycles, the PCI controller translates the device number to AD $n$  as shown in Table 17-50.

Table 17-50. PCI Type 0 Configuration—Device Number to AD $n$  Translation

Device Number		AD $n$ Used for IDSEL	Device Number		AD $n$ Used for IDSEL
Binary	Decimal		Binary	Decimal	
0b0_0000	0	— <sup>1</sup>	0b1_0100	20	AD20
0b0_0001–0b0_1001	1–9	— <sup>2</sup>	0b1_0101	21	AD21
0b0_1010	10	AD31	0b1_0110	22	AD22
0b0_1011	11	AD11	0b1_0111	23	AD23
0b0_1100	12	AD12	0b1_1000	24	AD24
0b0_1101	13	AD13	0b1_1001	25	AD25
0b0_1110	14	AD14	0b1_1010	26	AD26
0b0_1111	15	AD15	0b1_1011	27	AD27
0b1_0000	16	AD16	0b1_1100	28	AD28
0b1_0001	17	AD17	0b1_1101	29	AD29
0b1_0010	18	AD18	0b1_1110	30	AD30
0b1_0011	19	AD19	0b1_1111 <sup>3</sup>	31	—

<sup>1</sup>No external configuration transaction takes place; rather, internal registers are accessed.

<sup>2</sup>No IDSEL line asserted. Type0 configuration transaction is run, but ends with a master abort since no device responds.

<sup>3</sup>A device number of all ones indicates a PCI special-cycle or interrupt-acknowledge transaction.

For PCI type 0 translations, the function number and register number fields are copied without modification onto the PCI\_AD[10:2] signals during the address phase. The PCI\_AD[1:0] signals are

driven to 0b00 during the address phase for type 0 configuration cycles. The PCI controller implements address stepping on configuration cycles so that the target's IDSEL, which is connected directly to one of the PCI\_AD lines, reaches a stable value. This means that a valid address and command are driven on PCI\_AD[31:0] and PCI\_C/ $\overline{\text{BE}}$ [3:0] one clock cycle before the assertion of PCI\_FRAME.

#### 17.4.2.11.5 Type 1 Configuration Translation

For type 1 translations, the PCI controller copies the 30 high-order bits of the PCI\_CFG\_ADDR register (without modification) onto the PCI\_AD[31:2] signals during the address phase. The PCI controller automatically translates PCI\_AD[1:0] into 0b01 during the address phase to indicate a type 1 configuration cycle.

#### 17.4.2.12 Other Bus Transactions

There are two other PCI transactions that the PCI controller supports—interrupt acknowledge and special cycles. As an initiator, the PCI controller may initiate both interrupt acknowledge and special-cycle transactions; however, as a target, the PCI controller ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the PCI\_CFG\_ADDR and PCI\_CFG\_DATA registers described in [Section 17.4.2.11.3, “Agent Accessing the PCI Configuration Space.”](#)

##### 17.4.2.12.1 Interrupt-Acknowledge Transactions

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read operation implicitly addressed to the system interrupt controller. Note that the PCI interrupt-acknowledge command does not address the device's PIC processor interrupt-acknowledge register and does not return the interrupt vector address from the PIC unit. See [Chapter 10, “Programmable Interrupt Controller,”](#) for more information about the PIC unit.

When the PCI controller detects a read to the PCI\_CFG\_DATA register, it checks the enable flag and the device number in the PCI\_CFG\_ADDR register. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1\_1111), the function number is all ones (0b111), and the register number is zero (0b00\_0000), then the PCI controller performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the PCI controller performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

The address phase contains no valid information other than the interrupt-acknowledge command (PCI\_C/ $\overline{\text{BE}}$ [3:0] = 0b0000). Although there is no explicit address, PCI\_AD[31:0] are driven to a stable state, and parity is generated. Only one device (the system interrupt controller) on the PCI bus should respond to the interrupt-acknowledge command by asserting  $\overline{\text{PCI\_DEVSEL}}$ . All other devices on the bus should ignore the interrupt-acknowledge command. As stated previously, the device's PIC unit does not respond to PCI interrupt-acknowledge commands.

During the data phase, the responding device returns the interrupt vector on PCI\_AD[31:0] when PCI\_TRDY is asserted. The size of the interrupt vector returned is indicated by the value driven on the PCI\_C/ $\overline{\text{BE}}$ [3:0] signals.

The PCI controller also provides a direct way to generate PCI interrupt-acknowledge transactions. Reads from PCI INT\_ACK at offset 0x008 generate PCI interrupt-acknowledge transactions. Note that processor writes to these addresses do nothing.

#### 17.4.2.12.2 Special-Cycle Transactions

The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. The special-cycle command contains no explicit destination address but is broadcast to all PCI agents.

When the PCI controller detects a write to PCI\_CFG\_DATA, it checks the enable flag and the device number in PCI\_CFG\_ADDR. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1\_1111), the function number is all ones (0b111), and the register number is zero (0b00\_0000), then the PCI controller performs a special-cycle transaction on the local PCI bus. If the bus number indicates a nonlocal PCI bus, the PCI controller performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

Aside from the special-cycle command (PCI\_C/BE[3:0] = 0b0001) the address phase contains no other valid information. Although there is no explicit address, PCI\_AD[31:0] are driven to a stable state, and parity is generated. During the data phase, PCI\_AD[31:0] contain the special-cycle message and an optional data field. The special-cycle message is encoded on the 16 least-significant bits (PCI\_AD[15:0]); the optional data field is encoded on the most-significant 16 lines (PCI\_AD[31:16]). The special-cycle message encodings are assigned by the PCI SIG steering committee. The current list of defined encodings are provided in [Table 17-51](#).

**Table 17-51. Special-Cycle Message Encodings**

PCI_AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 architecture-specific
0x0003–0xFFFF	—

Note that the PCI controller does not automatically issue a special-cycle message when it enters any of its power-saving modes. It is the responsibility of software to issue the appropriate special-cycle message, if needed.

Each receiving agent must determine whether the special-cycle message is applicable to itself. Assertion of PCI\_DEVSEL in response to a special-cycle command is not necessary. The initiator of the special-cycle transaction can insert wait states but since there is no specific target, the special-cycle message and optional data field are valid on the first clock PCI\_IRDY is asserted. All special-cycle transactions are terminated by master-abort; however, the master-abort bit in the initiator's bus status register is not set for special-cycle terminations.

### 17.4.2.13 PCI Error Functions

PCI provides for parity and other system errors to be detected and reported. The PCI command register provides for selective enabling of specific PCI error detection. The PCI bus status register provides PCI error reporting. This section describes generation and detection of parity and error reporting for the PCI bus.

#### 17.4.2.13.1 PCI Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate even parity; that is, the number of ones on the PCI\_AD[31:0], PCI\_C/BE[3:0], and PCI\_PAR signals all sum to an even number.

Parity provides a way to determine, on each transaction, if the initiator successfully addressed the target and transferred valid data. The PCI\_C/BE[3:0] signals are included in the parity calculation to ensure that the correct bus command is performed (during the address phase) and correct data is transferred (during the data phase). The agent responsible for driving the bus must also drive even parity on the PAR and PCI\_PAR64 signal one clock cycle after a valid address phase or valid data transfer, as shown in Figure 17-60.

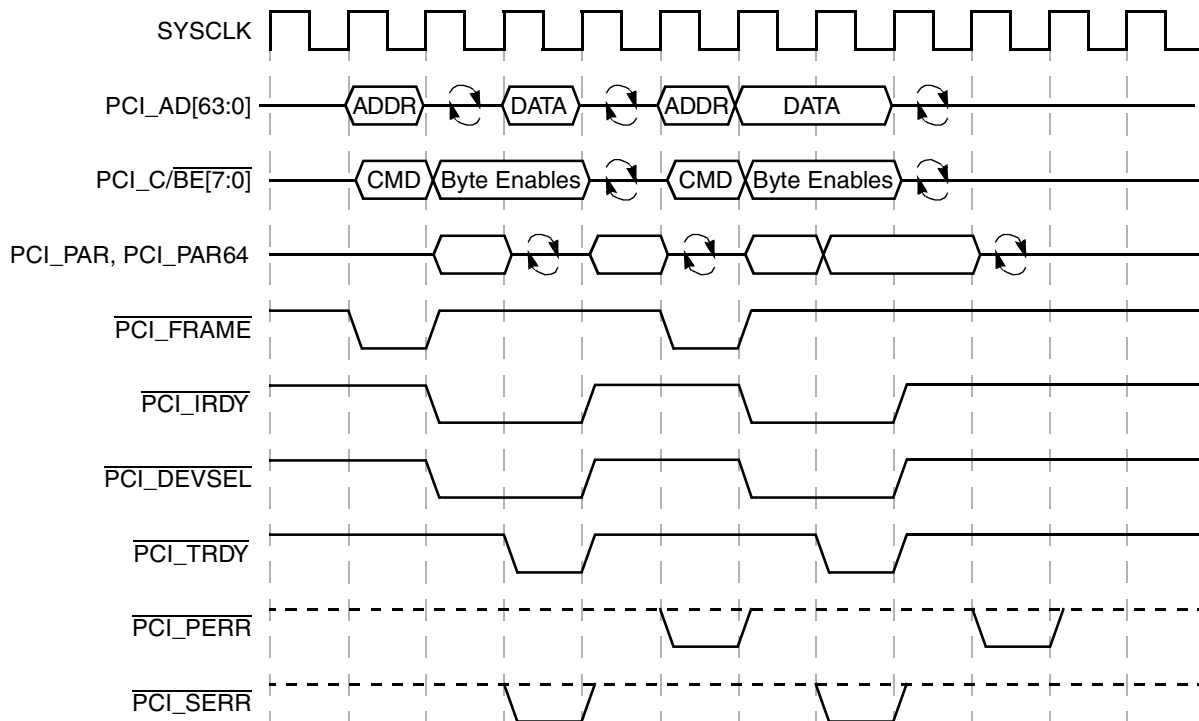


Figure 17-60. PCI Parity Operation

During the address and data phases, parity covers all 32 address/data signals and 4 command/byte enable signals, regardless of whether all lines carry meaningful information. Byte lanes not actually transferring data must contain stable (albeit meaningless) data and are included in parity calculation. During configuration, special-cycle, or interrupt-acknowledge commands; some address lines are not defined, but are driven to stable values and are included in parity calculation.

Agents that support parity checking must set the detected parity error bit in the PCI bus status register when a parity error is detected. Any additional response to a parity error is controlled by the parity error response bit in the PCI command register. If the parity error response bit is cleared, the agent ignores all parity errors.

### 17.4.2.13.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors— $\overline{\text{PCI\_PERR}}$  and  $\overline{\text{PCI\_SERR}}$ . The  $\overline{\text{PCI\_PERR}}$  signal is used exclusively to report data parity errors on all transactions except special cycles. The  $\overline{\text{PCI\_SERR}}$  signal is used for other error signaling including address parity errors and data parity errors on special-cycle transactions; it may also be used to signal other system errors.

Table 17-52 shows the actions taken for each kind of error.

**Table 17-52. PCI Mode Error Actions**

PCI Error Type	Error Detect Register bit	PCI Status Register bit	Comment
<b>PCI Outbound Read</b>			
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$	—	No data transferred
Received Parity Error for data phase	Mstr $\overline{\text{PERR}}$	Detected Parity Error, Master Data Parity Error Detected	No data transferred
Master Abort	Mstr abort	Received Master Abort	No data transferred
Target Abort	Trgt abort	Received Target Abort	No data transferred
Memory space violation	ORMSV	—	No data transferred. Only 8 bytes are requested in PCI bus
<b>PCI Outbound Write</b>			
Received $\overline{\text{SERR}}$ related to Address phase	Rcvd $\overline{\text{SERR}}$	—	May float AD bus to avoid contention
Received $\overline{\text{SERR}}$ related to Data phase	Rcvd $\overline{\text{SERR}}$	—	
Received $\overline{\text{PERR}}$ (Data phase)	Mstr $\overline{\text{PERR}}$	Master Data Parity Error	
Master Abort	Mstr abort	Received Master Abort	
Target Abort	Trgt abort	Received Target Abort	
Memory space violation	OWMSV	—	Only 8 bytes transferred.
<b>PCI Inbound Read</b>			
Detected Parity Error for Address phase	Addr Parity Error	Detected Parity Error, Signaled System Error	Float AD bus



Table 17-52. PCI Mode Error Actions (continued)

PCI Error Type	Error Detect Register bit	PCI Status Register bit	Comment
Detected Parity Error on upper address bus for Address phase (SAC or DAC)	—	—	
Received $\overline{\text{SERR}}$ at any phase	Received $\overline{\text{SERR}}$	—	
Received $\overline{\text{PERR}}$ (Data phase)	Target $\overline{\text{PERR}}$	—	
Internal error	Target Abort	Signaled Target Abort	
<b>PCI Inbound Write</b>			
Detected Parity Error for Address phase	Addr Parity Error	Detected Parity Error, Signaled System Error	Cache line purged
Detected Parity Error on upper address bus for Address phase (SAC or DAC)	—	—	
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$	—	
Detected Parity Error for Data phase	Trgt $\overline{\text{PERR}}$	Detected Parity Error	Cache line purged

## 17.5 Initialization/Application Information

This section describes some tips for use of the PCI controller.

### 17.5.1 Power-On Reset Configuration Modes

The PCI controller can power-on in three modes: host mode, agent mode and agent configuration lock mode. Certain bits in the configuration registers are set differently according to the POR (power-on reset) mode. Also, certain configuration bits have different implications when compared with past Freescale parts and PCI implementations. Note that after reset, the device cannot be switched from one mode to another.

The affected configuration bits are defined in [Table 17-53](#).

Table 17-53. Affected Configuration Register Bits for POR

Register (offset)	Bit	Name	Register Description
PCI Command Register (0x04)	2	Bus master	Controls whether the device can master a transaction on the PCI bus. If cleared, the device can not master a transaction. This bit is independent of host or agent mode.
	1	Memory space	Controls the acknowledgement of inbound memory transactions. If cleared, all inbound memory accesses (including accesses to PCSRBAR space) end in a master abort. This bit is independent of host or agent mode.

**Table 17-53. Affected Configuration Register Bits for POR (continued)**

Register (offset)	Bit	Name	Register Description
PCI Bus Function Register (0x44)	5	ACL	Valid only in agent mode. Controls acknowledgement of inbound configuration accesses. If set, all inbound configuration accesses are retried. If cleared, inbound configuration accesses are acknowledged. In host mode all inbound configuration accesses end in master aborts.
	0	PAH	Determines whether the device is in agent or host mode. Zero indicates host mode.

The POR reset values for the affected configuration bits are described in [Table 17-54](#).

**Table 17-54. Power-On Reset Values for Affected Configuration Bits**

Mode	Configuration Bit			
	Bus Master	Memory Space	ACL	PAH
Host	1	0	X	0
Agent	0	0	0	1
Agent configuration lock	0	0	1	1

### 17.5.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). The PBFR[ACL] bit is a don't care. The device powers up with the ability to master transactions on the PCI bus, however in order to acknowledge memory transactions, the memory space bit must be set.

### 17.5.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. However the device cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

### 17.5.1.3 Agent Configuration Lock Mode

Agent configuration lock mode is similar to agent mode with the added restriction that when the device powers up in agent configuration lock mode, it retries all inbound configuration accesses until the PBFR[ACL] bit is cleared. The purpose of this mode is to allow initial configuration on the port by the local processor before opening the port to be further configured by the external host. As in agent mode, the device in agent configuration lock mode cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

## 17.5.2 Byte Ordering

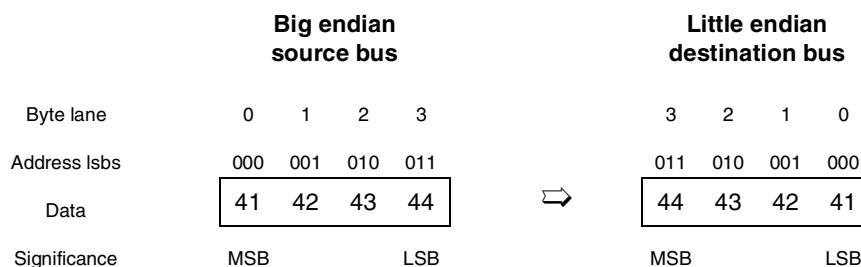
Whenever data must cross a bridge between two busses, the byte ordering of data on the source and destination busses must be considered. The internal platform bus of this device is inherently big endian and the PCI bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge—address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its byte ordering policy.

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register. This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

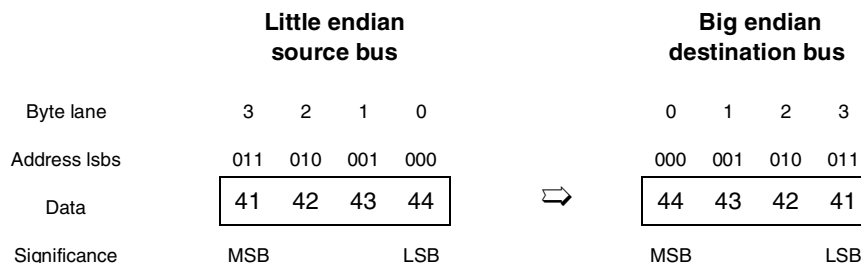
Figure 17-61 shows the transfer of a 4-byte scalar, 0x4142\_4344, from a big endian source across an address invariant bridge to a little endian destination.



**Figure 17-61. Address Invariant Byte Ordering—4 bytes Outbound**

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

Figure 17-63 shows data flowing the other way, from a little endian source to a big endian destination.



**Figure 17-62. Address Invariant Byte Ordering—4 bytes Inbound**

Figure 17-63 shows an outbound transfer of an 8-byte scalar, 0x5455\_1617\_CDCE\_2728, using address invariance.

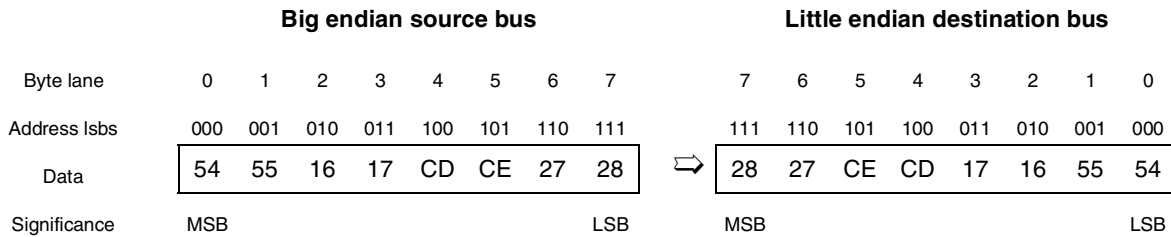


Figure 17-63. Address Invariant Byte Ordering—8 bytes Outbound

Figure 17-64 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

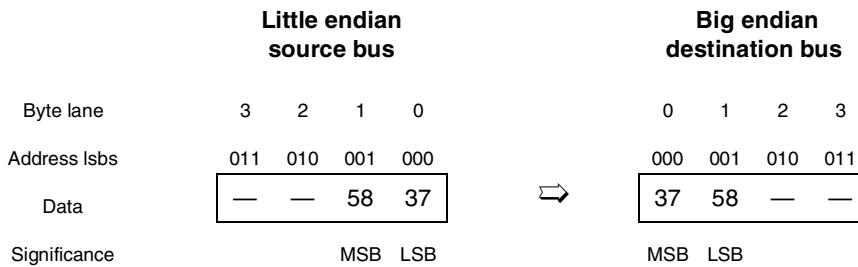


Figure 17-64. Address Invariant Byte Ordering—2 bytes Inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

### 17.5.2.1 Byte Order for Configuration Transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI specification defines PCI configuration registers as little endian. All accesses to the PCI configuration port, CFG\_DATA, including the those targeting the internal PCI configuration registers, use the address invariance policy as shown in Figure 17-65. Therefore, software must access CFG\_DATA with little-endian formatted data—either by using the `lwbrx/stwbrx` instructions or by manipulating the data before writing to and after reading from CFG\_DATA.

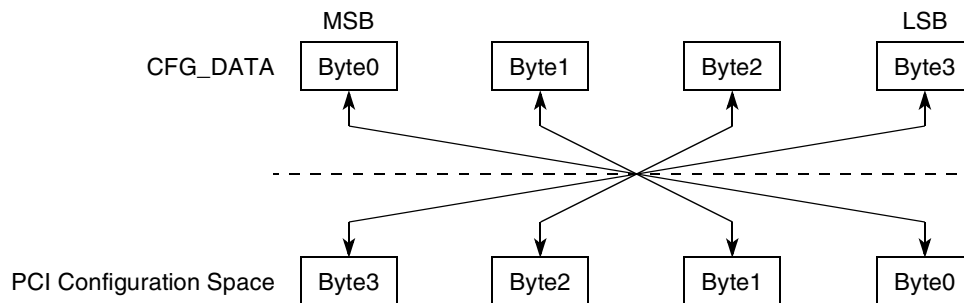


Figure 17-65. CFG\_DATA Byte Ordering

# Chapter 18

## PCI Express Interface Controller

The PCI Express interface complies with the *PCI Express™ Base Specification, Revision 1.0a* (available from <http://www.pcisig.org>). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol. The specific emphasis is directed at how the device implements the PCI Express specification. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express.

### NOTE

Much of the available PCI Express literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Note that this is inconsistent with the terminology in the rest of this manual where the terms ‘word’ and ‘double word’ refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

## 18.1 Introduction

The PCI Express controller provides the mechanism to communicate with PCI Express devices. [Figure 18-1](#) is a high-level block diagram of the PCI Express controller.

### 18.1.1 Overview

The PCI Express controller connects the internal platform to a 2.5-GHz serial interface. The MPC8533E offers three instantiations of this controller yielding up to two x4 links supported on SerDes 1 and a single x1 link on SerDes 2. The remainder of this chapter refers to a single PCI Express controller offering up to a x8 link interface. Notes are included to indicate variations for multiple instantiations.

As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. Upon coming out of reset, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner. Once link autonegotiation is successful, the controller is in operation.

Internally, the design contains queues to keep track of inbound and outbound transactions. There is control logic that handles buffer management, bus protocol, transaction spawning and tag generation. In addition, there are memory blocks used to store inbound and outbound data.

The PCI Express controller can be configured to operate as either a PCI Express root complex (RC) or an endpoint (EP) device. An RC device connects the host CPU/memory subsystem to I/O devices while an

EP device typically denotes a peripheral or I/O device. In RC mode, a PCI Express type 1 configuration header is used; in EP mode, a PCI Express type 0 configuration header is used.

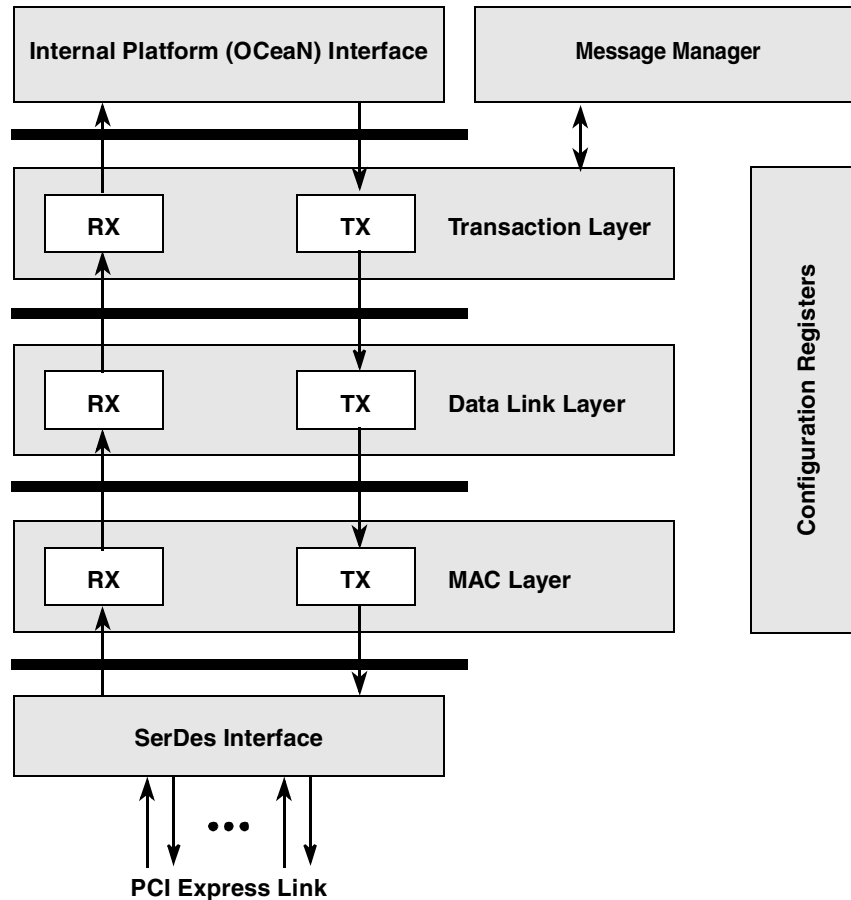


Figure 18-1. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. In addition, configuration and I/O transactions are supported if the PCI Express controller is in RC mode. As a target interface, the PCI Express controller accepts read and write operations to local memory space. When configured as an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers. Message generation and acceptance are supported in both RC and EP modes. Locked transactions and inbound I/O transactions are not supported.

### 18.1.1.1 Outbound Transactions

Outbound internal platform transactions to PCI Express are first mapped to a translation window to determine what PCI Express transactions are to be issued. A transaction from the internal platform can become a PCI Express Memory, I/O, Message, or Configuration transaction depending on the window attributes.

A transaction may be broken up into smaller sized transactions depending on the original request size, transaction type, and either the PCI Express device control register [MAX\_PAYLOAD\_SIZE] field for write requests or the PCI Express device control register [MAX\_READ\_SIZE] field for read requests. The controller performs PCI Express ordering rule checking to determine which transaction is to be sent on the PCI Express link.

In general, transactions are serviced in the order that they are received from the internal platform (OCeaN). Only when there is a stalled condition does the controller apply PCI Express ordering rules to outstanding transactions. For posted write transactions, once all data has been received from the internal platform (OCeaN), the data is forwarded to the PCI Express link and the transaction is considered as done. For non-posted write transactions, the controller waits for the completion packets to return before considering the transaction finished. For non-posted read transactions, the controller waits for all completion packets to return and then forwards all data back to the internal platform before terminating the transaction.

### 18.1.1.2 Inbound Transactions

Inbound PCI Express transactions to internal platform are first mapped to a translation window to determine what internal platform transactions are to be issued.

A transaction may be broken up into smaller sized transactions when sending to the internal platform depending on the original request size, byte enables and starting/ending addresses. The controller performs PCI Express ordering rule checking to determine what transaction is to be sent next to the internal platform (OCeaN).

In general, transactions are serviced in the order that they are received from the PCI Express link. Only when there is a stalled condition does the controller apply PCI Express ordering to outstanding transactions. For posted write transactions, once all data has been received from the PCI Express link, the data is forwarded to the internal platform and the transaction is considered as done. For non-posted read transactions, the controller forwards internal platform data back to the PCI Express link.

Note that the controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

## 18.1.2 Features

The following is a list of features supported by the PCI Express controller:

- Complies with the *PCI Express™ Base Specification, Revision 1.0a*
- Supports root complex (RC) and endpoint (EP) configurations
- 32- and 64-bit address support
- x4, x2 and x1 link support on SerDes 1; x1 link support on SerDes 2
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes
- Supports strong and relaxed transaction ordering rules
- PCI Express configuration registers (type 0 in EP mode, type 1 in RC mode)
- Baseline and advanced error reporting support

- One virtual channel (VC0)
- 256-byte maximum payload size (MAX\_PAYLOAD\_SIZE)
- Supports three inbound general-purpose translation windows and one configuration window
- Supports four outbound translation windows and one default window
- Supports eight non-posted and four posted PCI Express transactions
- Supports up to six priority 0 internal platform reads and eight priority 0 to 2 internal platform writes. (The maximum number of outstanding transactions at any given time is eight.)
- Credit-based flow control management
- Supports PCI Express messages and interrupts
- Accepts up to 256-byte transactions from the internal platform (OCeaN)

### 18.1.3 Modes of Operation

Several parameters that affect the PCI Express controller modes of operation are determined at power-on reset (POR) by reset configuration signals as described in [Chapter 4, “Reset, Clocking, and Initialization.”](#)

**Table 18-1. POR Parameters for PCI Express Controller**

Parameter	Description	Section/Page
Host/Agent Configuration	Selects between root complex (RC) and endpoint (EP) modes.	<a href="#">4.4.3.5/4-14</a>
I/O Port Selection	Selects the width of the PCI Express link	<a href="#">4.4.3.6/4-14</a>

#### 18.1.3.1 Root Complex/Endpoint Modes

The PCI Express controller can function as either a root complex (RC) or an endpoint (EP) on the PCI Express link. The host/agent configuration input signals `cfg_host_agt[0:2]` multiplexed on  $\overline{\text{LWE}}/\overline{\text{LBS}}[1:3]$  determine the RC/EP mode.

#### 18.1.3.2 Link Width

The I/O port selection configuration input signals `cfg_IO_ports[0:2]` multiplexed on `TSEC3_TXD[6:4]` determine the initial link width. (See [Section 4.4.3.6, “I/O Port Selection.”](#))

See [Section 4.4.4.2.1, “Minimum Frequency Requirements,”](#) for proper selection of CCB clock frequency with regard to PCI Express link width selection.

## 18.2 External Signal Descriptions

Although the generic PCI Express controller described here accommodates up to a single x8 link, the three PCI Express controller instantiations implemented on the MPC8533E offer either dual x4 links and a single x1 link, or dual x2 links and a single x4 link. Please refer to [Section 4.4.3.6, “I/O Port Selection,”](#) and the *MPC8533EEC Electrical Characteristics* document for specific pin muxing details.



PCI Express defines the connection between two devices as a link, which can be composed of a single or multiple lanes. Each lane consists of a differential pair for transmitting ( $TX_n$  and  $\overline{TX}_n$ ) and a differential pair for receiving ( $RX_n$  and  $\overline{RX}_n$ ) with an embedded data clock.

Table 18-2 contains detailed descriptions of the external PCI Express interface signals.

**Table 18-2. PCI Express Interface Signals—Detailed Signal Descriptions**

Signal	I/O	Description
SD1_RX[7:0] SD2_RX0	I	Receive data. The receive data signals carry PCI Express packet information. Subject to <a href="#">Section 4.4.3.6, “I/O Port Selection</a> , PCI Express controller 1 is available on SD1_RX[3:0], PCI Express controller 2 is available on SD1_RX[7:4], and PCI Express controller 3 is available on SD2_RX0.
		<b>State Meaning</b> Asserted/Negated—Represents data being received from the PCI Express interface.
		<b>Timing</b> Assertion/Negation—As described in the <i>PCI Express Base Specification, Revision 1.0a</i> .
$\overline{SD1\_RX}$ [7:0] $\overline{SD2\_RX0}$	I	Receive data, inverted. $\overline{SD1\_RX}$ [7:0] and $\overline{SD2\_RX0}$ are the inverted forms of the receive data signals (SD1_RX[7:0] and SD2_RX0).
		<b>State Meaning</b> Asserted/Negated—Represents the inverse of data being received from the PCI Express interface.
		<b>Timing</b> Assertion/Negation—As described in the <i>PCI Express Base Specification, Revision 1.0a</i> .
SD1_TX[7:0] SD2_TX0	O	Transmit data. The transmit data signals carry PCI Express packet information. Subject to <a href="#">Section 4.4.3.6, “I/O Port Selection</a> , PCI Express controller 1 is available on SD1_TX[3:0], PCI Express controller 2 is available on SD1_TX[7:4], and PCI Express controller 3 is available on SD2_TX0.
		<b>State Meaning</b> Asserted/Negated—Represents data being transmitted to the PCI Express interface.
		<b>Timing</b> Assertion/Negation—As described in the <i>PCI Express Base Specification, Revision 1.0a</i> .
$\overline{SD1\_TX}$ [7:0] $\overline{SD2\_TX0}$	O	Transmit data, inverted. $\overline{SD1\_TX}$ [7:0] and $\overline{SD2\_TX0}$ are the inverted form of the transmit data signals (SD1_TX[7:0] and SD2_TX0).
		<b>State Meaning</b> Asserted/Negated—Represents the inverse of data being transmitted to the PCI Express interface.
		<b>Timing</b> Assertion/Negation—As described in the <i>PCI Express Base Specification, Revision 1.0a</i> .

## 18.3 Memory Map/Register Definitions

The PCI Express interface supports the following register types:

- Memory-mapped registers—these registers control PCI Express address translation, PCI error management, and PCI Express configuration register access. These registers are described in [Section 18.3.1, “PCI Express Memory Mapped Registers,”](#) and its subsections.
- PCI Express configuration registers contained within the PCI Express configuration space—these registers are specified by the PCI Express specification for every PCI Express device. These registers are described in [Section 18.3.7, “PCI Express Configuration Space Access,”](#) and its subsections.

### 18.3.1 PCI Express Memory Mapped Registers

The PCI Express memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PEXCSRBAR on the PCI Express

side) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI Express configuration data register, PEX\_CONFIG\_DATA) must only be accessed as 32-bit quantities.

Also note that although the table explicitly lists only the registers for the PCI Express Controller 1, the register map for PCI Express Controllers 2 and 3 are the same except for the block base address. Memory-mapped registers for PCI Express Controller 1 begin at block base address 0x0\_A000, controller 2 registers begin at 0x0\_9000, and controller 3 registers begin at 0x0\_B000.

Table 18-3 lists the memory-mapped registers. In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 18-3. PCI Express Memory-Mapped Register Map**

Offset	Register	Access	Reset	Section/Page
<b>PCI Express Controller 1 Memory-Mapped Registers—Block Base Address 0x0_A000</b>				
<b>PCI Express Configuration Access Registers</b>				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	<a href="#">18.3.2.1/18-9</a>
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	<a href="#">18.3.2.2/18-10</a>
0x008	Reserved	—	—	
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	<a href="#">18.3.2.3/18-11</a>
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	<a href="#">18.3.2.4/18-11</a>
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	<a href="#">18.3.2.5/18-12</a>
0x018–0x01C	Reserved	—	—	
<b>PCI Express Power Management Event &amp; Message Registers</b>				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	<a href="#">18.3.3.1/18-13</a>
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	<a href="#">18.3.3.2/18-14</a>
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	<a href="#">18.3.3.3/18-16</a>
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	<a href="#">18.3.3.4/18-17</a>
0x030–0xBF4	Reserved	—	—	
<b>PCI Express IP Block Revision Registers</b>				

Table 18-3. PCI Express Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0100	<a href="#">18.3.4.1/18-18</a>
0xBF8	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	<a href="#">18.3.4.2/18-19</a>
<b>PCI Express ATMU Registers</b>				
<b>Outbound Window 0 (Default)</b>				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC08–0xC0C	Reserved	—	—	
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC14–0xC1C	Reserved	—	—	
<b>Outbound Window 1</b>				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC2C	Reserved	—	—	
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC34–0xC3C	Reserved	—	—	
<b>Outbound Window 2</b>				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC4C	Reserved	—	—	
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC54–0xC5C	Reserved	—	—	
<b>Outbound Window 3</b>				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC6C	Reserved	—	—	

Table 18-3. PCI Express Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC74– 0xC7C	Reserved	—	—	
<b>Outbound Window 4</b>				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC8C	Reserved	—	—	
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC94– 0xC9C	Reserved	—	—	
0xCA0– 0xD9C	Reserved	—	—	
<b>Inbound Window 3</b>				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDA4	Reserved	—	—	
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.5/18-27</a>
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>
0xDB4– 0xDBC	Reserved	—	—	
<b>Inbound Window 2</b>				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDC4	Reserved	—	—	
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.5/18-27</a>
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>
0xDD4– 0xDDC	Reserved	—	—	
<b>Inbound Window 1</b>				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDE4	Reserved	—	—	
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDEC	Reserved	—	—	
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>

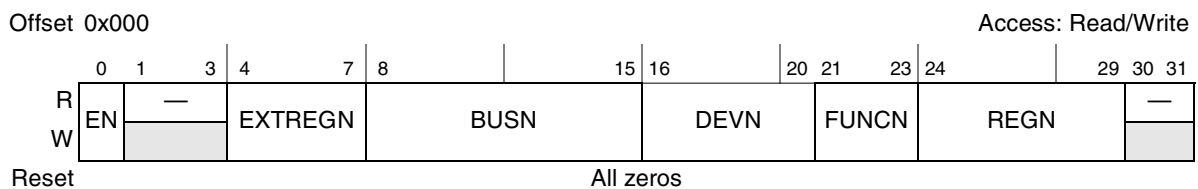
**Table 18-3. PCI Express Memory-Mapped Register Map (continued)**

Offset	Register	Access	Reset	Section/Page
0xDF4–0xDFC	Reserved	—	—	
<b>PCI Express Error Management Registers</b>				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	<a href="#">18.3.6.1/18-29</a>
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	<a href="#">18.3.6.2/18-32</a>
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	<a href="#">18.3.6.3/18-34</a>
0xE14–0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	<a href="#">18.3.6.4/18-35</a>
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	<a href="#">18.3.6.5/18-36</a>
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	<a href="#">18.3.6.6/18-38</a>
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	<a href="#">18.3.6.7/18-39</a>
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	<a href="#">18.3.6.8/18-40</a>
0xE38–0xFFC	Reserved	—	—	
<b>PCI Express Controller 2 Memory-Mapped Registers—Block Base Address 0x0_9000</b>				
0x000–0xFFC	PCI Express Controller 2 registers <b>Note:</b> All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 2; the offsets of PCI Express Controller 2 registers are the same except they have a different block base address.			
<b>PCI Express Controller 3 Memory-Mapped Registers—Block Base Address 0x0_B000</b>				
0x000–0xFFC	PCI Express Controller 3 registers <b>Note:</b> All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 3; the offsets of PCI Express Controller 3 registers are the same except they have a different block base address.			

## 18.3.2 PCI Express Configuration Access Registers

### 18.3.2.1 PCI Express Configuration Address Register (PEX\_CONFIG\_ADDR)

The PCI Express configuration address register, shown in [Figure 18-2](#), contains address information for accesses to PCI Express internal and external configuration registers.



**Figure 18-2. PCI Express Configuration Address Register (PEX\_CONFIG\_ADDR)**

The fields of the PCI Express configuration address register are described in [Table 18-4](#).

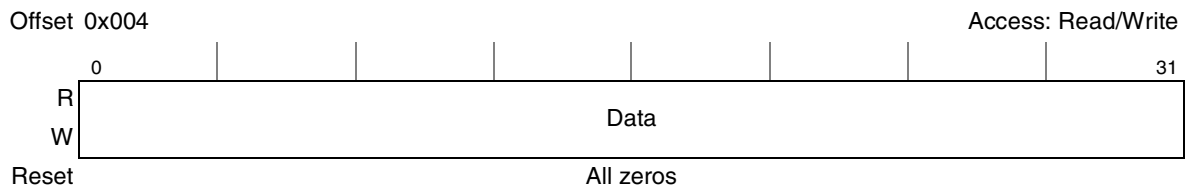
**Table 18-4. PEX\_CONFIG\_ADDR Field Descriptions**

Bits	Name	Description
0	EN	Enable. This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed. If this bit is cleared, writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data.
1–3	—	Reserved
4–7	EXTREGN	Extended register number. This field allows access to extended PCI Express configuration space (that is, the registers in the offset range from 0x100 to 0xFFFF).
8–15	BUSN	Bus number. PCI bus number to access
16–20	DEVN	Device number. Device number to access on specified bus
21–23	FUNCN	Function number. Function to access within specified device
24–29	REGN	Register number. 32-bit register to access within specified device
30–31	—	Reserved

Both root complex (RC) and endpoint (EP) configuration headers contain 4096 bytes of address space. To access a register within the header, both the extended register number and the register number fields are concatenated to form the 4-byte aligned address of the register. That is, the register address is extended register number || register number || 0b00.

### 18.3.2.2 PCI Express Configuration Data Register (PEX\_CONFIG\_DATA)

The PCI Express configuration data register, show in [Figure 18-3](#), is a 32-bit port for internal and external configuration access. Note that accesses of 1, 2, or 4 bytes to the PCI Express configuration data register are allowed. Also note that accesses to the little-endian PCI Express configuration space must be properly formatted. See [Section 18.4.1.2.1, “Byte Order for Configuration Transactions,”](#) for more information.



**Figure 18-3. PCI Express Configuration Data Register (PEX\_CONFIG\_DATA)**

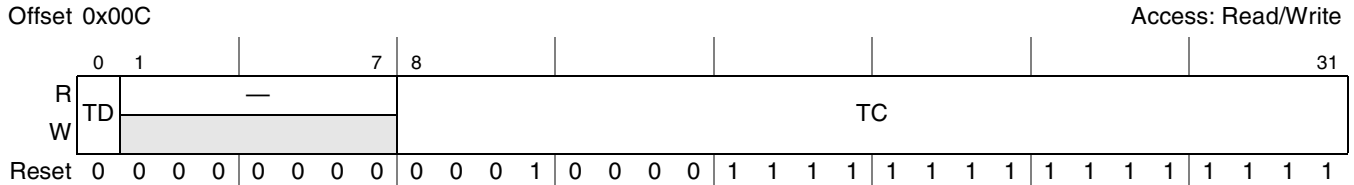
The fields of the PCI Express configuration data register are described in [Table 18-5](#).

**Table 18-5. PEX\_CONFIG\_DATA Field Descriptions**

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI Express configuration cycle if the PEX_CONFIG_ADDR enable bit is set (PEX_CONFIG_ADDR[EN] = 1).

### 18.3.2.3 PCI Express Outbound Completion Timeout Register (PEX\_OTB\_CPL\_TOR)

The PCI Express outbound completion timeout register, shown in [Figure 18-4](#), contains the maximum wait time for a response to come back as a result of an outbound non-posted request before a timeout condition occurs.



**Figure 18-4. PCI Express Outbound Completion Timeout Register (PEX\_OTB\_CPL\_TOR)**

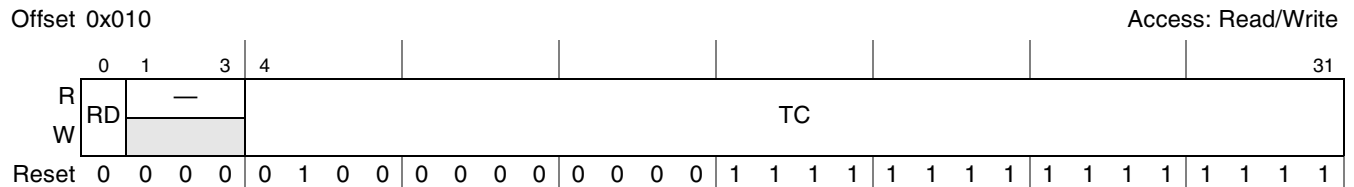
The fields of the PCI Express outbound completion timeout register are described in [Table 18-6](#).

**Table 18-6. PEX\_OTB\_CPL\_TOR Field Descriptions**

Bits	Name	Description
0	TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
1–7	—	Reserved
8–31	TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. One TC unit is 24 ns at 333 MHz and 30 ns at 260 MHz. Timeout period based on different TC settings: 0x00_0000 Reserved 0x10_FFFF 26.72 ms at 333.33 MHz platform clock 0xFF_FFFF 446.94 ms at 333.33 MHz platform clock

### 18.3.2.4 PCI Express Configuration Retry Timeout Register (PEX\_CONF\_RTY\_TOR)

The PCI Express configuration retry timeout register, shown in [Figure 18-5](#), contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.



**Figure 18-5. PCI Express Configuration Retry Timeout Register (PEX\_CONF\_RTY\_TOR)**

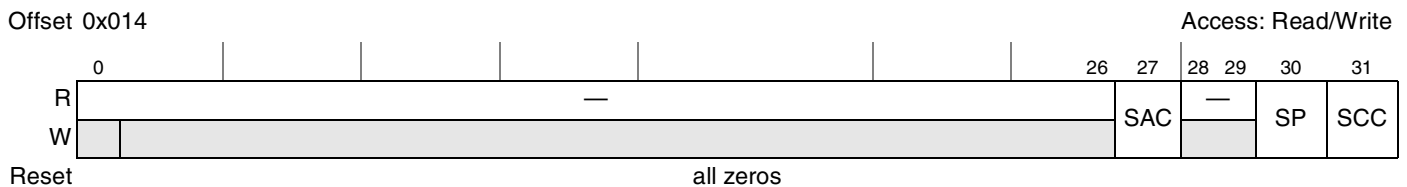
The fields of the PCI Express configuration retry timeout register are described in [Table 18-7](#).

**Table 18-7. PEX\_CONF\_RTY\_TOR Field Descriptions**

Bits	Name	Description
0	RD	Retry disable. This bit disables the retry of a configuration transaction that receives a CRS status response packet. 0 Enable retry of a configuration transaction in response to receiving a CRS status response until the timeout counter (defined by the PEX_CONF_RTY_TOR[TC] field) has expired. 1 Disable retry of a configuration transaction regardless of receiving a CRS status response.
1–3	—	Reserved
4–31	TC	Timeout counter. This is the value that is used to load the CRS response counter. One TC unit is 24 ns at 333 MHz and 30 ns at 260 MHz. Timeout period based on different TC settings: 0x000_0000 Reserved 0x400_FFFF 1.612 seconds at 333.33 MHz platform clock 0xFFF_FFFF 7.12 seconds at 333.33 MHz platform clock

### 18.3.2.5 PCI Express Configuration Register (PEX\_CONFIG)

The PCI Express configuration register, shown in [Figure 18-6](#), contains various control switches for the controller.



**Figure 18-6. PCI Express Configuration Register (PEX\_CONFIG)**

The fields of the PCI Express configuration register are described in [Table 18-8](#).

**Table 18-8. PEX\_CONFIG Field Descriptions**

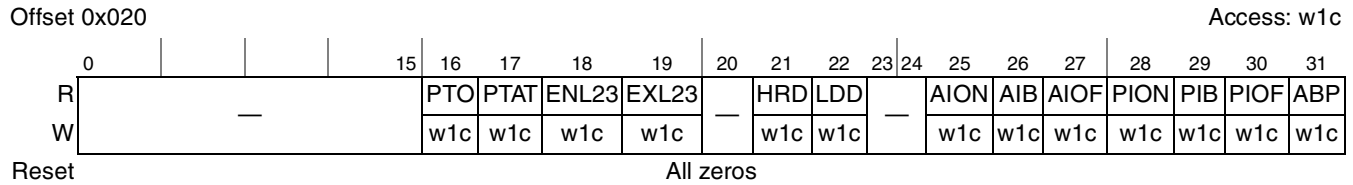
Bits	Name	Description
0–26	—	Reserved
27	SAC	Sense ASPM Control. This bit controls the default value of ASPM of PEX Link Control Register's bit 0. See <a href="#">Section 18.3.9.11, "PCI Express Link Control Register—0x5C,"</a> for more information.
28–29	—	Reserved
30	SP	Slot Present. This bit controls the default value of the PCI Express capabilities register [slot] bit. See <a href="#">Section 18.3.9.6, "PCI Express Capabilities Register—0x4E,"</a> for more information.
31	SCC	Slot Clock Configuration. This bit controls the default value of the PCI Express link status register [SCC] bit. See <a href="#">Section 18.3.9.12, "PCI Express Link Status Register—0x5E,"</a> for more information.



## 18.3.3 PCI Express Power Management Event and Message Registers

### 18.3.3.1 PCI Express PME and Message Detect Register (PEX\_PME\_MES\_DR)

The PCI Express PME and message detect register, shown in [Figure 18-7](#), logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.



**Figure 18-7. PCI Express PME and Message Detect Register (PEX\_PME\_MES\_DR)**

The fields of the PCI Express PME and message detect register are described in [Table 18-9](#).

**Table 18-9. PEX\_PME\_MES\_DR Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16	PTO	PME turn off. This bit indicates the detection of a PME_Turn_Off message. This bit is only valid in EP mode. 1 A PME_Turn_Off message is detected 0 No PME_Turn_Off message detected
17	PTAT	PME to ack time-out. This bit indicates the detection of a PME_to_ack time-out condition. This bit is only valid in RC mode. 1 A PME_TO_Ack time-out condition is detected 0 No PME_TO_Ack time-out condition detected
18	ENL23	Entered L2/L3 ready state. This bit indicates that the PCI Express controller has entered L2/L3 state. This is only valid in RC mode. 1 L2/L3 ready state has been entered 0 L2/L3 ready state has not been entered
19	EXL23	Exit L2/L3 ready state. This bit indicates that the PCI Express controller has exited the L2/L3 state. This is only valid in RC mode. 1 Exit L2/L3 state has been detected 0 Exit L2/L3 state not detected
20	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
21	HRD	Hot reset detected. This bit indicates that the PCI Express controller has detected a hot reset condition on the link. The controller will be reset and will clean up all outstanding transactions. Link retraining will take place once hot reset state is exited. This is valid only in EP mode. 1 Hot reset request has been detected 0 Hot reset request not detected
22	LDD	Link down detected. This bit indicates that a link down condition has been detected. The controller will be reset and then will clean up all outstanding transactions. Link retraining will take place once the controller has cleaned itself up. 1 Link down has been detected 0 Link down not detected
23–24	—	Reserved

**Table 18-9. PEX\_PME\_MES\_DR Field Descriptions (continued)**

Bits	Name	Description
25	AION	Attention indicator on. This bit indicates the detection of an Attention_Indicator_On message. This bit is only valid in EP mode. 1 Attention indicator on message is detected 0 No attention indicator on message detected
26	AIB	Attention indicator blink. This bit indicates the detection of an Attention_Indicator_Blink message. This bit is only valid in EP mode. 1 Attention indicator blink message is detected 0 No attention indicator blink message detected
27	AIOF	Attention indicator off. This bit indicates the detection of an Attention_Indicator_Off message. This bit is only valid in EP mode. 1 Attention indicator off message is detected 0 No attention indicator off message detected
28	PION	Power indicator on. This bit indicates the detection of a Power_Indicator_On message. This bit is only valid in EP mode. 1 Power indicator on message is detected 0 No power indicator on message detected
29	PIB	Power indicator blink. This bit indicates the detection of an Power_Indicator_Blink message. This bit is only valid in EP mode. 1 Power indicator blink message is detected 0 No power indicator blink message detected
30	PIOF	Power indicator off. This bit indicates the detection of an Power_Indicator_Off message. This bit is only valid in EP mode. 1 Power indicator off message is detected 0 No power indicator off message detected
31	ABP	Attention button pressed. This bit indicates the detection of an Attention_Button_Pressed message. This bit is only valid in EP mode. 1 Attention button press message is detected 0 No attention button press message detected

### 18.3.3.2 PCI Express PME and Message Disable Register (PEX\_PME\_MES\_DISR)

The PCI Express PME and message disable register, shown in [Figure 18-8](#), when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.



**Figure 18-8. PCI Express PME and Message Disable Register (PEX\_PME\_MES\_DISR)**

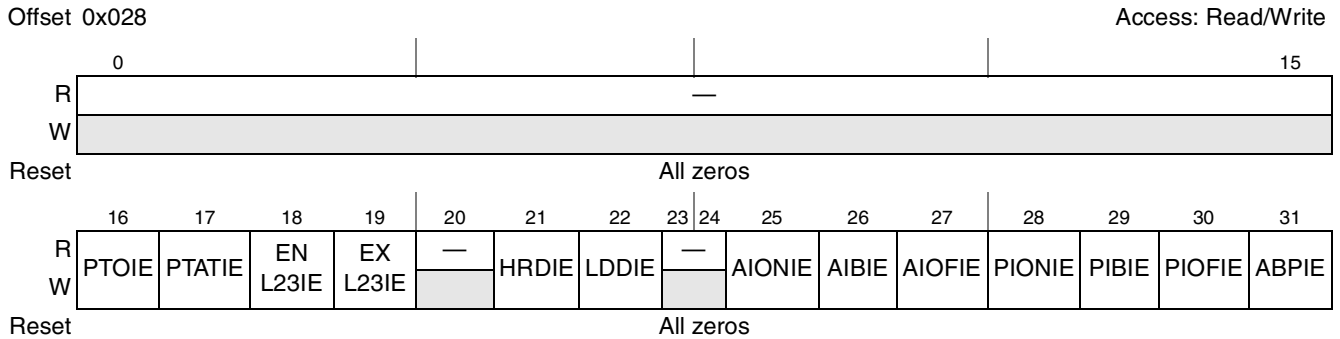
The fields of the PCI Express PME and message disable register are described in [Table 18-10](#).

**Table 18-10. PEX\_PME\_MES\_DISR Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16	PTOD	PME turn off disable. When set will disable the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
17	PTATD	PME to ack time-out disable. When set will disable the setting of PEX_PME_MES_DR[PTAT] bit. 1 Disable PME_TO_Ack time-out detection 0 Enable PME_TO_Ack time-out detection
18	ENL23D	Entered_L2/L3 ready disable. When set will disable the setting of PEX_PME_MES_DR[ENL23] bit. 1 Disable Entered_L2/L3 ready state detection 0 Enable Entered_L2/L3 ready state detection
19	EXL23D	Exited_L2/L3 ready disable. When set will disable the setting of PEX_PME_MES_DR[EXL23] bit. 1 Disable Exited_L2/L3 ready state detection 0 Enable Exited_L2/L3 ready state detection
20	—	Reserved
21	HRDD	Hot reset detected disable. When set will disable the setting of PEX_PME_MES_DR[HRD] bit. 1 Disable hot reset state detection 0 Enable hot reset state detection
22	LDDD	Link down detected disable. When set will disable the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
23–24	—	Reserved
25	AIOND	Attention indicator on disable. When set will disable the setting of PEX_PME_MES_DR[AION] bit. 1 Disable attention indicator on message detection 0 Enable attention indicator on message detection
26	AIBD	Attention indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[AIB] bit. 1 Disable attention indicator blink message detection 0 Enable attention indicator blink message detection
27	AIOFD	Attention indicator off disable. When set will disable the setting of PEX_PME_MES_DR[AIOF] bit. 1 Disable attention indicator off message detection 0 Enable attention indicator off message detection
28	PIOND	Power indicator on disable. When set will disable the setting of PEX_PME_MES_DR[PION] bit. 1 Disable power indicator on message detection 0 Enable power indicator on message detection
29	PIBD	Power indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[PIB] bit. 1 Disable power indicator blink message detection 0 Enable power indicator blink message detection
30	PIOFD	Power indicator off disable. When set will disable the setting of PEX_PME_MES_DR[PIOF] bit. 1 Disable power indicator off message detection 0 Enable power indicator off message detection
31	ABPD	Attention button pressed disable. When set will disable the setting of PEX_PME_MES_DR[ABP] bit. 1 Disable attention button press message detection 0 Enable attention button press message detection

### 18.3.3.3 PCI Express PME and Message Interrupt Enable Register (PEX\_PME\_MES\_IER)

The PCI Express PME and message interrupt enable register, shown in Figure 18-9, allows for the detection of a message or a PME event to generate an interrupt, provided that the corresponding bit in the PCI Express PME and message detect register is set.



**Figure 18-9. PCI Express PME and Message Interrupt Enable Register (PEX\_PME\_MES\_IER)**

Table 18-11 shows the fields of the PCI Express PME and message interrupt enable register.

**Table 18-11. PEX\_PME\_MES\_IER Field Descriptions**

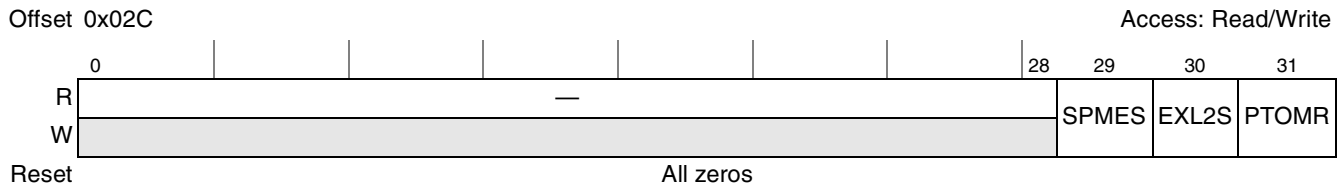
Bits	Name	Description
0–15	—	Reserved
16	PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 will generate an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
17	PTATIE	PME To ack time-out interrupt enable. When set and PEX_PME_MES_DR[PTAT]=1 will generate an interrupt. 1 Enable PME_TO_Ack time-out interrupt generation 0 Disable PME_TO_Ack time-out interrupt generation
18	ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 will generate an interrupt. 1 Enable Entered_L2/L3 ready state interrupt generation 0 Disable Entered_L2/L3 ready state interrupt generation
19	EXL23IE	Exited L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[EXL23]=1 will generate an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
20	—	Reserved
21	HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 will generate an interrupt. 1 Enable hot reset state interrupt generation 0 Disable hot reset state interrupt generation
22	LDDIE	Link down detected interrupt enable. When set and PEX_PME_MES_DR[LDD]=1 will generate an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation

**Table 18-11. PEX\_PME\_MES\_IER Field Descriptions (continued)**

Bits	Name	Description
23–24	—	Reserved
25	AIONIE	Attention indicator on interrupt enable. When set and PEX_PME_MES_DR[AION]=1 will generate an interrupt. 1 Enable attention indicator on message interrupt generation 0 Disable attention indicator on message interrupt generation
26	AIBIE	Attention indicator blink interrupt enable. When set and PEX_PME_MES_DR[AIB]=1 will generate an interrupt. 1 Enable attention indicator blink message interrupt generation 0 Disable attention indicator blink message interrupt generation
27	AIOFIE	Attention indicator off interrupt enable. When set and PEX_PME_MES_DR[AIOF]=1 will generate an interrupt. 1 Enable attention indicator off message interrupt generation 0 Disable attention indicator off message interrupt generation
28	PIONIE	Power indicator on interrupt enable. When set and PEX_PME_MES_DR[PION]=1 will generate an interrupt. 1 Enable power indicator on message interrupt generation 0 Disable power indicator on message interrupt generation
29	PIBIE	Power indicator blink interrupt enable. When set and PEX_PME_MES_DR[PIB]=1 will generate an interrupt. 1 Enable power indicator blink message interrupt generation 0 Disable power indicator blink message interrupt generation
30	PIOFIE	Power indicator off interrupt enable. When set and PEX_PME_MES_DR[PIOF]=1 will generate an interrupt. 1 Enable power indicator off message interrupt generation 0 Disable power indicator off message interrupt generation
31	ABPIE	Attention button pressed interrupt enable. When set and PEX_PME_MES_DR[ABP]=1 will generate an interrupt. 1 Enable attention button press message interrupt generation 0 Disable attention button press message interrupt generation

### 18.3.3.4 PCI Express Power Management Command Register (PEX\_PMCR)

The PCI Express power management command register, shown in [Figure 18-10](#), provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

**Figure 18-10. PCI Express Power Management Command Register (PEX\_PMCR)**

The fields of the PCI Express power management command register are described in [Table 18-12](#).

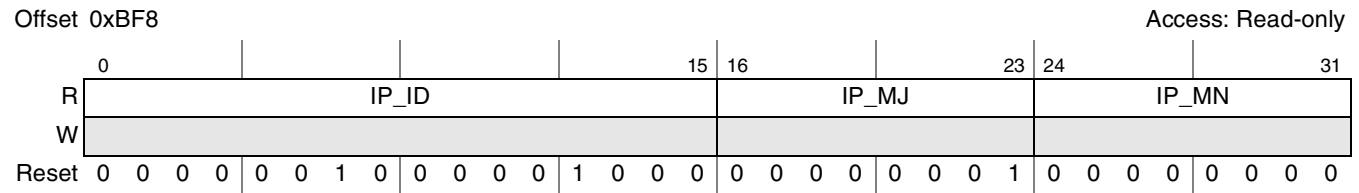
**Table 18-12. PEX\_PMCR Field Descriptions**

Bits	Name	Description
0–28	—	Reserved
29	SPMES	Set PME status. This will set the PME status bit and if PME is enabled (see <a href="#">Section 18.3.9.3, “PCI Express Power Management Status and Control Register—0x48,”</a> on page 18-69 for more information) it will transmit a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
30	EXL2S	Exit L2 state. When set will exit the link state out of L2/L3 ready state in order to send new requests. The request is only made when entered L2/L3 ready state is active. This bit is self-clearing. When the link has exited L2/L3 ready state, the status bit Exit_L2/L3 ready state will be set. This bit should not be used when in EP mode.
31	PTOMR	PME_Turn_Off message request. When set will broadcast a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

### 18.3.4 PCI Express IP Block Revision Registers

#### 18.3.4.1 IP Block Revision Register 1 (PEX\_IP\_BLK\_REV1)

The IP block revision register 1 is shown in [Figure 18-11](#).



**Figure 18-11. IP Block Revision Register 1**

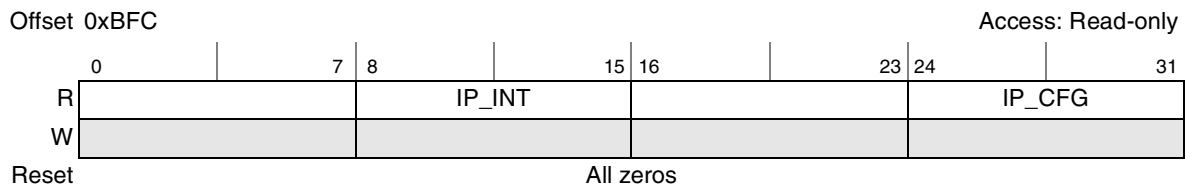
[Table 18-13](#) contains descriptions of the fields of the IP block revision register 1.

**Table 18-13. PCI Express IP Block Revision Register 1 Field Descriptions**

Bits	Name	Description
0–15	IP_ID	Block ID
16–23	IP_MJ	Block Major Revision
24–31	IP_MN	Block Minor Revision

### 18.3.4.2 IP Block Revision Register 2 (PEX\_IP\_BLK\_REV2)

The IP block revision register 2 is shown in [Figure 18-12](#).



**Figure 18-12. IP Block Revision Register 2**

[Table 18-14](#) contains descriptions of the fields of the IP block revision register 2.

**Table 18-14. PCI Express IP Block Revision Register 2 Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	Block integration option
16–23	—	Reserved
24–31	IP_CFG	Block configuration option

## 18.3.5 PCI Express ATMU Registers

### 18.3.5.1 PCI Express Outbound ATMU Registers

The outbound address translation windows must be aligned based on the granularity selected by the size fields. Outbound window misses use the default outbound register set (outbound ATMU window 0). Overlapping outbound windows are not supported and will cause undefined behavior. Note that for RC mode, all outbound transactions post ATMU must hit either into the memory base/limit range or the prefetchable memory base/limit range defined in the PCI Express type 1 header. For EP mode, there is no such requirement.

Note that in RC mode, there is no checking on whether the translated address actually hits into the memory base/limit range. It will just pass it through as is.

Figure 18-13 shows the outbound transaction flow.

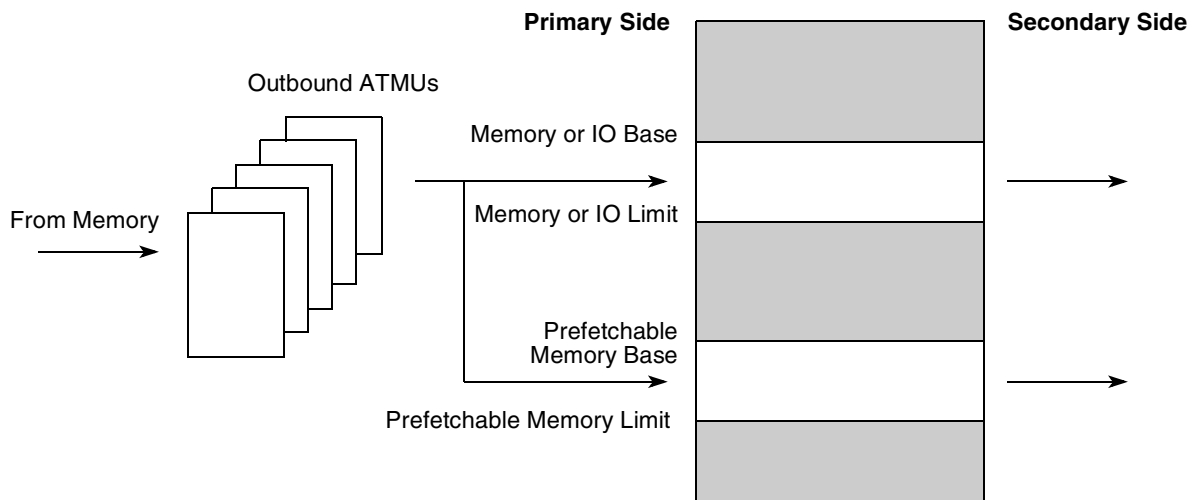


Figure 18-13. RC Outbound Transaction Flow

### 18.3.5.1.1 PCI Express Outbound Translation Address Registers (PEXOTAR<sub>n</sub>)

The PCI Express outbound translation address registers, shown in Figure 18-14, select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

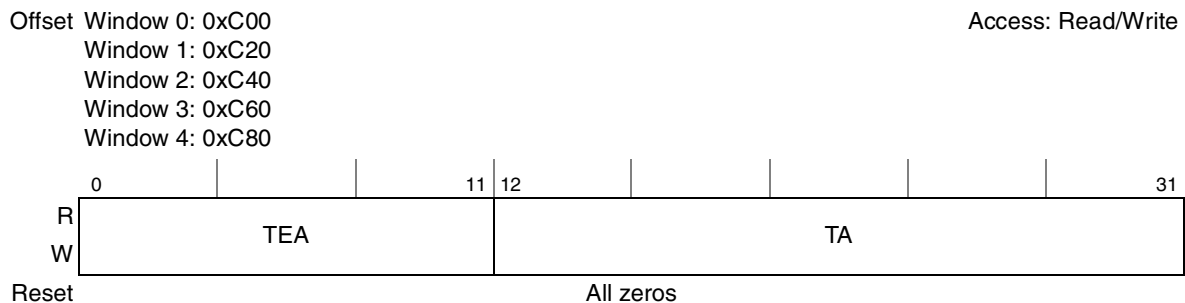


Figure 18-14. PCI Express Outbound Translation Address Registers (PEXOTAR<sub>n</sub>)

Table 18-15 describes the fields of the PCI Express outbound translation address registers.

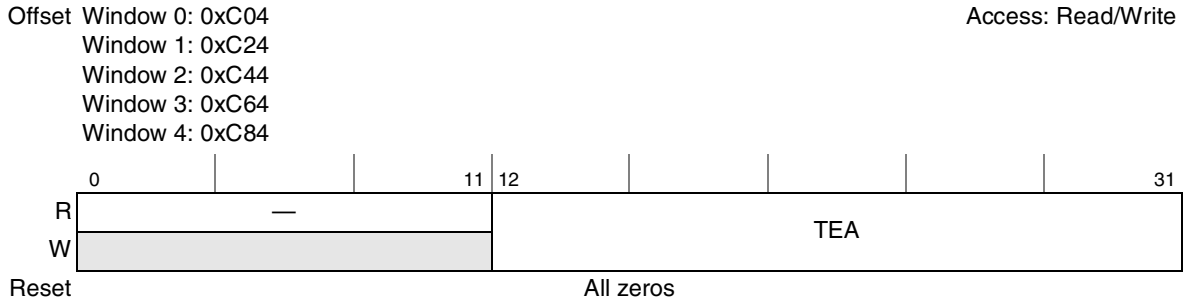
Table 18-15. PEXOTAR<sub>n</sub> Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [43:32] (bit 32 is the lsb).
12–31	TA	Translation address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to PCI Express address bits [31:12].



### 18.3.5.1.2 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR<sub>n</sub>)

The PCI Express outbound translation extended address registers, shown in Figure 18-15, contain the most-significant bits of a 64 bit translation address.



**Figure 18-15. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR<sub>n</sub>)**

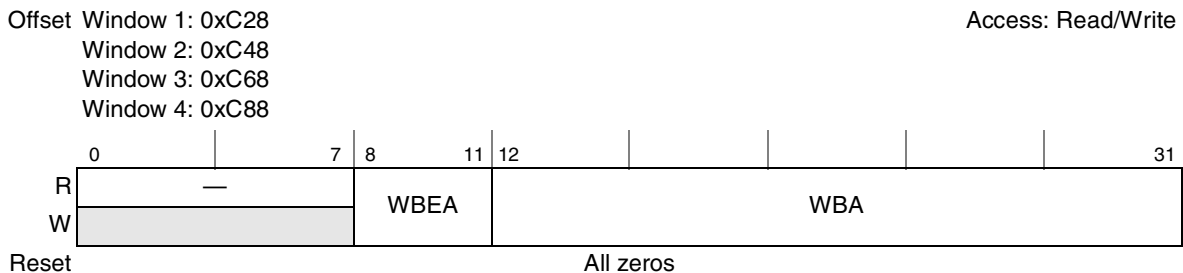
Table 18-16 describes the fields of the PCI Express outbound translation extended address registers.

**Table 18-16. PCI Express Outbound Extended Address Translation Register *n* Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [63:44].

### 18.3.5.1.3 PCI Express Outbound Window Base Address Registers (PEXOWBAR<sub>n</sub>)

The PCI Express outbound window base address registers, shown in Figure 18-16, select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set.



**Figure 18-16. PCI Express Outbound Window Base Address Registers (PEXOWBAR<sub>n</sub>)**

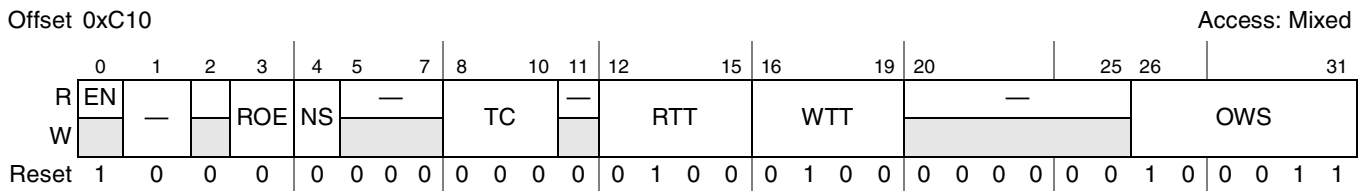
Table 18-17 describes the fields of the PCI Express outbound window base address registers.

**Table 18-17. PCI Express Outbound Window Base Address Register *n* Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–11	WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. Correspond to internal platform address bits [0:3]. (where 0 is the msb of the internal platform address)
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to internal platform address bits [4:23].

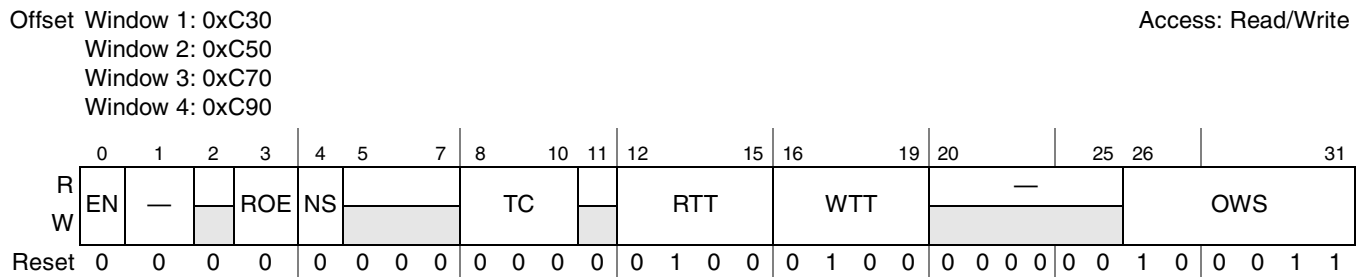
**18.3.5.1.4 PCI Express Outbound Window Attributes Registers (PEXOWAR<sub>*n*</sub>)**

The PCI Express outbound window attributes registers, shown in Figure 18-17 and Figure 18-18, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. Figure 18-17 shows the outbound window attributes register 0 (PEXOWAR<sub>0</sub>).



**Figure 18-17. PCI Express Outbound Window Attributes Register 0 (PEXOWAR<sub>0</sub>)**

Figure 18-18 shows the PCI Express outbound window attributes registers 1–4 (PEXOWAR<sub>*n*</sub>).



**Figure 18-18. PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR<sub>*n*</sub>)**

Table 18-18 describes the fields of the PCI Express outbound window attributes registers.

**Table 18-18. PEXOWAR<sub>*n*</sub> Field Descriptions**

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1–2	—	Reserved

Table 18-18. PEXOWAR<sub>n</sub> Field Descriptions (continued)

Bits	Name	Description
3	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set will enable the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set will enable the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7	—	Reserved
8–10	TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7 <b>Note:</b> Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class.
11	—	Reserved
12–15	RTT	Read transaction type. Read transaction type to run on the PCI Express link 0000 Reserved 0001 Reserved 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory read ... Reserved 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. ... Reserved 1111 Reserved
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0000 Reserved 0001 Reserved 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. Note that inbound write transactions on one PCI express port must not translate to outbound configuration write transactions on another PCI Express port. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. ... Reserved 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. Note that inbound write transactions on one PCI express port must not translate to outbound I/O write transactions on another PCI Express port. ... Reserved 1111 Reserved

Table 18-18. PEXOWAR $n$  Field Descriptions (continued)

Bits	Name	Description
20–25	—	Reserved
26–31	OWS	<p>Outbound window size. Outbound translation window size N which is the encoded <math>2^{(N + 1)}</math>-byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window will be aliased to the default window.</p> <p>000000 Reserved</p> <p>...</p> <p>001011 4-Kbyte window size</p> <p>001100 8-Kbyte window size</p> <p>...</p> <p>011111 4-Gbyte window size</p> <p>100000 8-Gbyte window size</p> <p>100001 16-Gbyte window size</p> <p>100010 32-Gbyte window size</p> <p>100011 64-Gbyte window size</p> <p>100100 Reserved</p> <p>...</p> <p>111111 Reserved</p>

### 18.3.5.2 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

#### 18.3.5.2.1 EP Inbound ATMU Implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible via PEX\_CONFIG\_ADDR/PEX\_CONFIG\_DATA mechanism. Note that host software must program these BAR using configuration type 0 cycles. There are 4 inbound BARs.

- Default inbound window BAR0 at configuration address 0x10 (32-bit). Also known as PEXCSRBAR. This is a fixed 1-Mbyte window used for inbound memory transactions that access memory-mapped registers.
- Inbound window BAR1 at configuration address 0x14 (32-bit)
- Inbound window BAR2 at configuration address 0x18-0x1c (64-bit)
- Inbound window BAR3 at configuration address 0x20-0x24 (64-bit)

The PCI Express controller does not implement a shadow of the inbound BARs in the memory-mapped register set. However, when there is a hit to the BAR(s), the PCI Express controller uses the corresponding translation and attribute registers in the memory-mapped register set for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR will be used.

### 18.3.5.2.2 RC Inbound ATMU Implementation

In RC mode, the PEXIWBAR[1–3] registers reside outside of the type 1 header; PEXIWBAR0 is the only inbound BAR that resides in the Type 1 header (at offset 0x10).

If the transaction hits any window, the translation is performed and then the transaction is sent to memory. If there is no hit to any one of the BARs, then a UR completion will be returned for non-posted transactions. All posted transactions with no BAR hit are ignored.

Figure 18-19 shows the inbound transaction flow in RC mode.

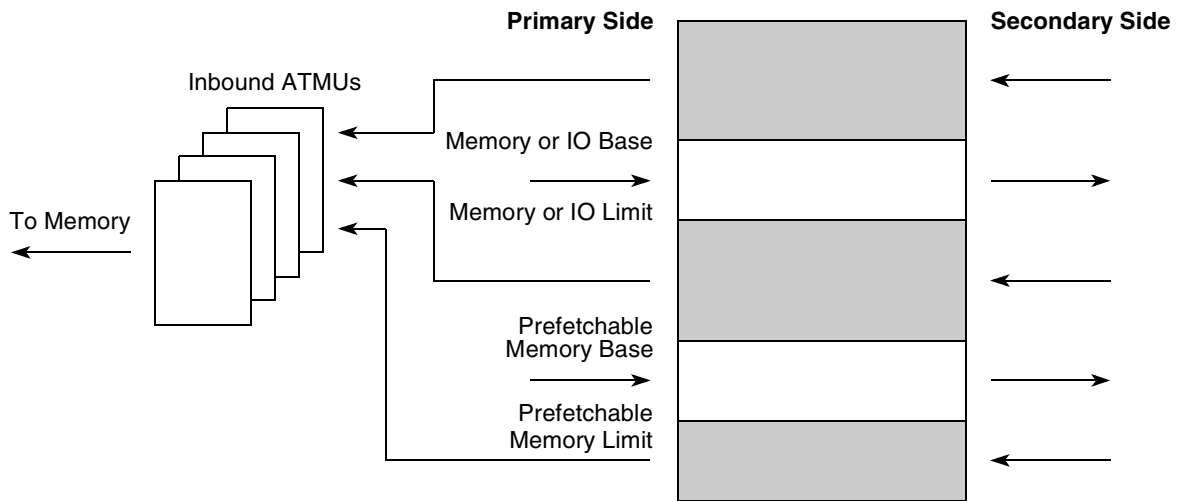


Figure 18-19. RC Inbound Transaction Flow

### 18.3.5.2.3 PCI Express Inbound Translation Address Registers (PEXITAR<sub>n</sub>)

The PCI Express inbound translation address registers, shown in Figure 18-20, contain the translated internal platform address to be used. Note that PEXITAR0 does not exist in the memory-mapped space; it is a fixed 1-Mbyte translation to the internal configuration (CCSRBAR) space.

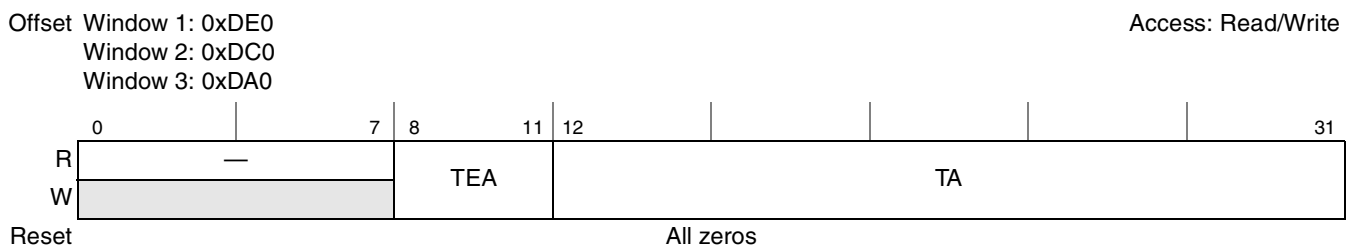


Figure 18-20. PCI Express Inbound Translation Address Registers (PEXITAR<sub>n</sub>)

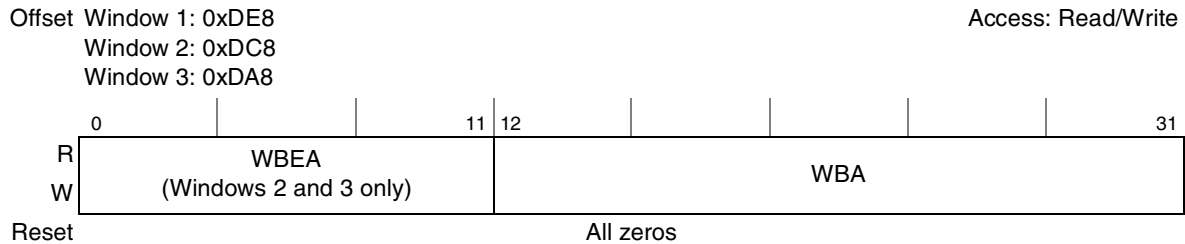
Table 18-19 describes the fields of the PCI Express inbound translation address registers.

**Table 18-19. PCI Express Inbound Translation Address Registers Field Descriptions**

Bits	Name	Description
0–7	—	Reserved
8–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:3] where bit 0 is the msb of the internal platform address.
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to internal platform address bits [4:23].

### 18.3.5.2.4 PCI Express Inbound Window Base Address Registers (PEXIWBAR<sub>n</sub>)

The PCI Express inbound window base address registers, shown in Figure 18-21, select the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for inbound transactions are compared to these windows. In RC mode, PEXIWBAR<sub>0</sub> is located in the PCI Express type 1 configuration header space and PEXIWBAR[1–3] registers are implemented as described in this section. In endpoint (EP) mode, these registers are not implemented in the memory-mapped space. Reading these registers in EP mode will return all zeros and writing to these offsets has no consequences. All base address registers in EP are located in the PCI Express type 0 configuration header space. Note that PEXIWBAR<sub>1</sub> only supports 32-bit PCI Express address space.



**Figure 18-21. PCI Express Inbound Window Base Address Registers (PEXIWBAR<sub>n</sub>)**

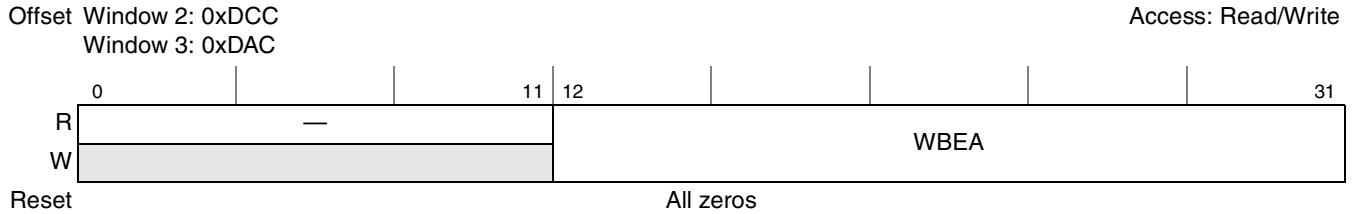
Table 18-20 describes the fields of the PCI Express inbound window base address registers.

**Table 18-20. PCI Express Inbound Window Base Address Register Field Descriptions**

Bits	Name	Description
0–11	WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32]. Note that the extended address is supported for windows 2 and 3 only; for PEXIWBAR <sub>1</sub> , these bits are reserved and must be 0.
12–31	WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

### 18.3.5.2.5 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEN<sub>n</sub>)

The PCI Express inbound window base extended address registers, shown in Figure 18-22, contain the most-significant bits of a 64 bit base address.



**Figure 18-22. PCI Express Inbound Window Base Extended Address Registers (PEXIWBEN<sub>n</sub>)**

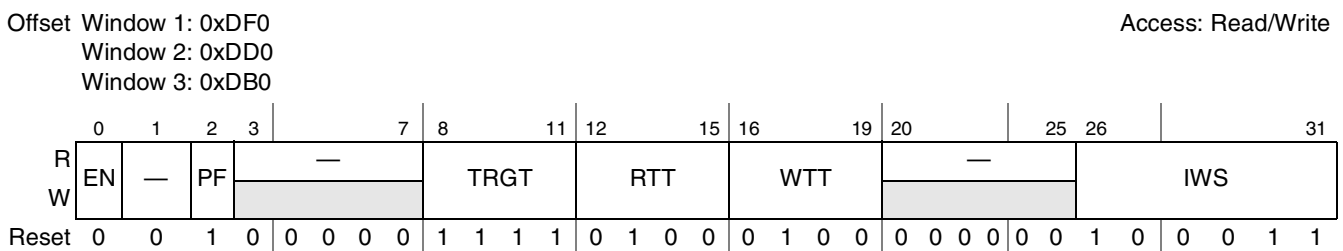
Table 18-21 describes the fields of the PCI Express inbound window base extended address registers.

**Table 18-21. PCI Express Inbound Window Base Extended Address Register Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12–31	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

### 18.3.5.2.6 PCI Express Inbound Window Attributes Registers (PEXIWAR<sub>n</sub>)

The PCI Express inbound window attributes registers, shown in Figure 18-23, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.



**Figure 18-23. PCI Express Inbound Window Attributes Registers (PEXIWAR<sub>n</sub>)**

Table 18-22 describes the fields of the PCI Express inbound window attributes registers.

**Table 18-22. PCI Express Inbound Window Attributes Registers Field Descriptions**

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the translation window. 0 Disable inbound window translation 1 Enable inbound window translation
1	—	Reserved

**Table 18-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)**

Bits	Name	Description
2	PF	<p>Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode.</p> <p>0 Not prefetchable 1 Prefetchable</p>
3–7	—	Reserved
8–11	TRGT	<p>Target interface. If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.</p> <p>0000 PCI 0001 PCI Express 2—PCI Express controller 2 should not use this encoding 0010 PCI Express 1—PCI Express controller 1 should not use this encoding 0011 PCI Express 3—PCI Express controller 3 should not use this encoding 0100-1110 Reserved 1111 Local memory space</p> <p><b>Note:</b> Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.</p>
12–15	RTT	<p>Read transaction type. Read transaction type to send to target interface.</p> <p>If the transaction is not going to local memory space</p> <p>0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved</p> <p>If the transaction is going to local memory space</p> <p>0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved</p>



**Table 18-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)**

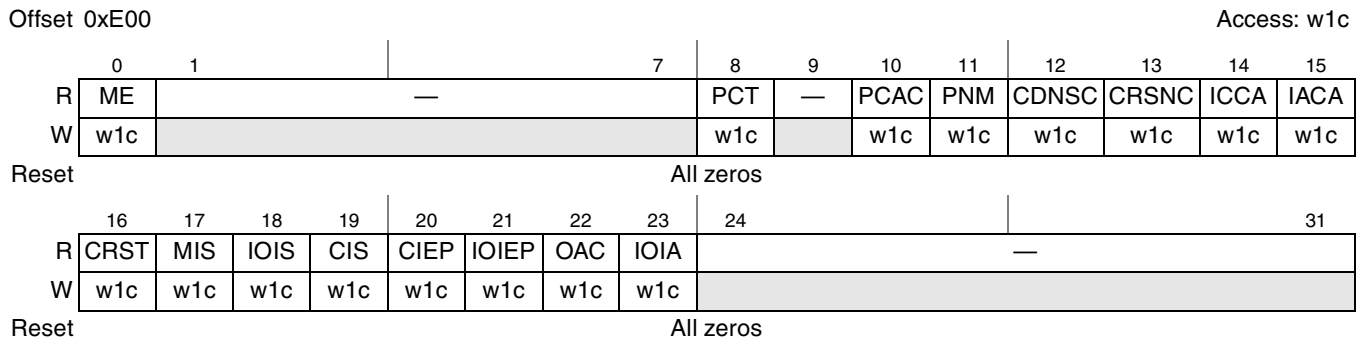
Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to send to target interface.  If the transaction is not going to local memory space  0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved  If the transaction is going to local memory space  0000 Reserved ... 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N+1)}$ -bytes window size. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved ... 111111 Reserved

## 18.3.6 PCI Express Error Management Registers

### 18.3.6.1 PCI Express Error Detect Register (PEX\_ERR\_DR)

The PCI Express error detect register, shown in [Figure 18-24](#), contains error status bits that are detected by hardware. The detected error bits are write-1-to-clear type registers. Reading from these registers occurs

normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0b0200\_0000 is written to the register. When an error is detected the appropriate error bit is set. Subsequent errors will set the appropriate error bits in the error detection registers, but only the first error for a particular unit will have any relevant information captured. The interrupt enable bits are used to allow or block the error reporting to the interrupt mechanism while the disable bits are used to prevent or allow the setting of the status bits.



**Figure 18-24. PCI Express Error Detect Register (PEX\_ERR\_DR)**

Table 18-23 describes the fields of the PCI Express error detect register.

**Table 18-23. PCI Express Error Detect Register Field Descriptions**

Bits	Name	Description
0	ME	Multiple errors. Detecting multiple errors of the same type. An error is considered as multiple error when its detect bit is set and the same error is occurring again. Note that this bit does not track the ordering of when the error occurs. 1 Multiple errors were detected. 0 Multiple errors were not detected.
1–7	—	Reserved
8	PCT	PCI Express completion time-out. A completion time-out condition was detected for a non-posted, outbound PCI Express transaction. An error response is sent back to the requestor. 1 A completion time-out on the PCI Express link was detected. 0 No completion time-out on the PCI Express link detected.
9	—	Reserved
10	PCAC	PCI Express completer abort (CA) completion. A completion with CA status was received. 1 A completion with CA status was detected. 0 No completion with CA status detected.
11	PNM	PCI Express no map. Detect an inbound transaction that was not mapped to any inbound windows. In RC mode, a completion without data (Cpl) packet with a UR completion status is sent back to the requester and this bit is set. For EP mode, a Cpl packet with a UR completion status is sent back to the requester but will not set this bit. 1 A no-map transaction was detected in RC mode. 0 No no-map transaction detected.

**Table 18-23. PCI Express Error Detect Register Field Descriptions (continued)**

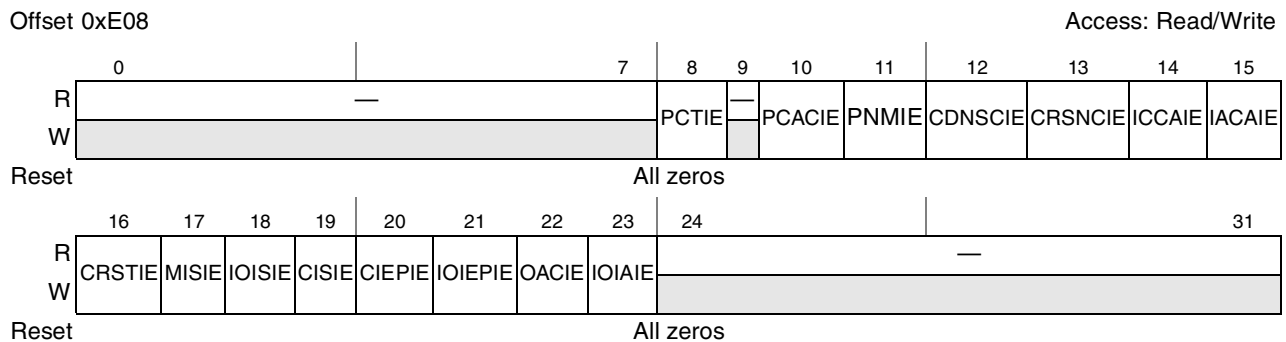
Bits	Name	Description
12	CDNSC	Completion with data not successful. A completion with data packet was received with a non successful status (i.e. UR, CA or CRS status). 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
13	CRSNC	CRS non configuration. A completion was detected for a non configuration cycle and with CRS status. 1 CRS non configuration packet was detected. 0 No CRS non configuration packet detected.
14	ICCA	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access. Access to an illegal configuration space from PEX_CONFIG_ADDR/PEX_CONFIG_DATA was detected. 1 Invalid CONFIG_ADDR/PEX_CONFIG_DATA access detected 0 No invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detected
15	IACA	Invalid ATMU configuration access. Access to an illegal configuration space from an ATMU window was detected. 1 Invalid ATMU configuration access was detected 0 No invalid ATMU configuration access detected
16	CRST	CRS thresholded. An outbound configuration transaction was retried and thresholded due to a CRS completion status. An error response is sent back to the requestor. See <a href="#">Section 18.3.2.4, "PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)"</a> for more information. 1 A CRS threshold condition was detected for an outbound configuration transaction 0 No CRS threshold condition detected
17	MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See <a href="#">Section 18.4.1.8.1, "Outbound ATMU Message Generation"</a> , for more information. 1 An invalid size outbound message transaction was detected 0 No invalid size outbound message transaction detected
18	IOIS	I/O invalid size. An outbound I/O transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 an invalid size outbound I/O transaction was detected 0 no invalid size outbound I/O transaction detected
19	CIS	Configuration invalid size. An outbound configuration transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound configuration transaction was detected 0 No invalid size outbound configuration transaction detected
20	CIEP	Configuration invalid EP. An outbound ATMU configuration transaction request was seen when in EP mode. 1 An outbound configuration transaction while in EP was detected 0 No outbound configuration transaction in EP detected
21	IOIEP	I/O invalid EP. An outbound I/O transaction request was seen when in EP mode. 1 An outbound I/O transaction while in EP was detected 0 No outbound I/O transaction in EP detected
22	OAC	Outbound ATMU crossing. An outbound crossing ATMU transaction was detected. 1 An outbound transaction that hits in one window and crosses overing it was detected 0 No outbound ATMU crossing condition detected

**Table 18-23. PCI Express Error Detect Register Field Descriptions (continued)**

Bits	Name	Description
23	IOIA	I/O invalid address. An outbound I/O transaction with a translated address of greater than 4 Gbytes was detected. 1 A greater than 4-Gbyte I/O address was detected 0 No greater than 4-Gbyte I/O address detected
24–31	—	Reserved

### 18.3.6.2 PCI Express Error Interrupt Enable Register (PEX\_ERR\_EN)

The PCI Express error interrupt enable register, shown in [Figure 18-25](#), allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.



**Figure 18-25. PCI Express Error Interrupt Enable Register (PEX\_ERR\_EN)**

[Table 18-24](#) describes the fields of the PCI Express error interrupt enable register.

**Table 18-24. PCI Express Error Interrupt Enable Register Field Descriptions**

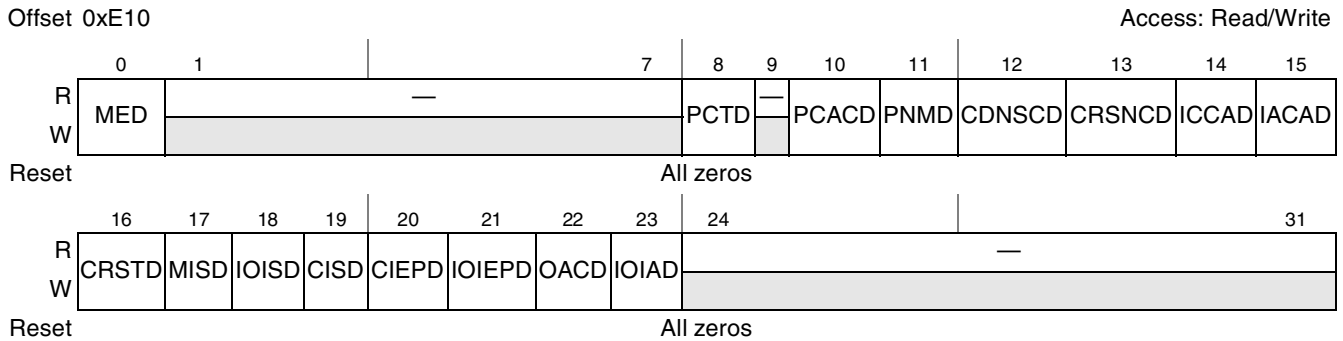
Bits	Name	Description
0–7	—	Reserved
8	PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 will generate an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
9	—	Reserved
10	PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 will generate an interrupt. 1 Enable completion with CA status interrupt generation 0 Disable completion with CA status interrupt generation
11	PNMIE	PCI Express no map interrupt enable. When set and PEX_ERR_DR[PNM]=1 will generate an interrupt. 1 Enable no map PCI Express packet interrupt generation 0 Disable no map PCI Express packet interrupt generation
12	CDNSCIE	Completion with data not successful interrupt enable. When this bit is set and PEX_ERR_DR[CDNSC] = 1 will generate an interrupt. 1 Enable completion with data non successful interrupt generation 0 Disable completion with data non successful interrupt generation

Table 18-24. PCI Express Error Interrupt Enable Register Field Descriptions (continued)

Bits	Name	Description
13	CRSNCIE	CRS non configuration interrupt enable. When this bit is set and PEX_ERR_DR[CRSNC] = 1 will generate an interrupt. 1 Enable CRS non configuration interrupt generation 0 Disable CRS non configuration interrupt generation
14	ICCAIE	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access interrupt enable. When set and PEX_ERR_DR[ICCA]=1 will generate an interrupt. 1 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation 0 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation.
15	IACAIE	Invalid ATMU configuration access. When set and PEX_ERR_DR[IACA]=1 will generate an interrupt. 1 Enable invalid ATMU configuration access interrupt generation 0 Disable invalid ATMU configuration access interrupt generation
16	CRSTIE	CRS thresholded interrupt enable. When set and PEX_ERR_DR[CRST]=1 will generate an interrupt. 1 Enable CRS threshold interrupt generation 0 Disable CRS threshold interrupt generation
17	MISIE	Message invalid size interrupt enable. When set and PEX_ERR_DR[MIS]=1 will generate an interrupt. 1 Enable invalid outbound message size interrupt generation 0 Disable invalid outbound message size interrupt generation
18	IOISIE	I/O invalid size interrupt enable. When set and PEX_ERR_DR[IOIS]=1 will generate an interrupt. 1 Enable invalid outbound I/O size interrupt generation 0 Disable invalid outbound I/O size interrupt generation
19	CISIE	Configuration invalid size interrupt enable. When set and PEX_ERR_DR[CIS]=1 will generate an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation
20	CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 will generate an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation
21	IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 will generate an interrupt. 1 Enable outbound I/O transaction EP mode interrupt generation 0 Disable outbound I/O transaction EP mode interrupt generation
22	OACIE	Outbound ATMU crossing interrupt enable. When set and PEX_ERR_DR[OAC]=1 will generate an interrupt. 1 Enable outbound crossing ATMU interrupt generation 0 Disable outbound crossing ATMU interrupt generation
23	IOIAIE	I/O address invalid enable. When set and PEX_ERR_DR[IOIA]=1 will generate an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
24–31	—	Reserved

### 18.3.6.3 PCI Express Error Disable Register (PEX\_ERR\_DISR)

The PCI Express error disable register, shown in Figure 18-26, controls the setting of the PCI Express error detect register’s bits.



**Figure 18-26. PCI Express Error Disable Register (PEX\_ERR\_DISR)**

Table 18-25 describes the fields of the PCI Express error disable register.

**Table 18-25. PCI Express Error Disable Register Field Descriptions**

Bits	Name	Description
0	MED	Multiple errors disable. When set will disable the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
1–7	—	Reserved
8	PCTD	PCI Express completion time-out disable. When set will disable the setting of PEX_ERR_DR[PET] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection
9	—	Reserved
10	PCACD	PCI Express CA completion disable. When set will disable the setting of PEX_ERR_DR[PCAC] bit. 1 Disable completion with CA status detection 0 Enable completion with CA status detection
11	PNMD	PCI Express no map disable. When set will disable the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection
12	CDNSCD	Completion with data not successful disable. When set will disable the setting of PEX_ERR_DR[CDNSC] bit. 1 Disable completion with data not successful detection 0 Enable completion with data not successful detection
13	CRSNCD	CRS non configuration disable. When set will disable the setting of PEX_ERR_DR[CRSNC] bit. 1 Disable CRS non configuration detection 0 Enable CRS non configuration detection
14	ICCAD	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access disable. When set will disable the setting of PEX_ERR_DR[ICCA] bit. 1 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection 0 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection



Table 18-26 describes the fields of the PCI Express error capture status register.

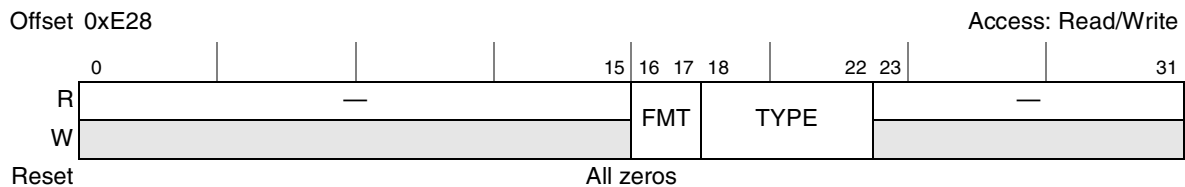
**Table 18-26. PCI Express Error Capture Status Register Field Descriptions**

Bits	Name	Description
0–24	—	Reserved
25	TO	Transaction originator. This field Indicates whether the originator of the transaction is from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 1 Transaction originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 0 Transaction not originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA.
26–30	GSID	Global source ID. This field indicates the internal platform global source ID that the error transaction originates. This field only applies to non PEX_CONFIG_ADDR/PEX_CONFIG_DATA transactions. 00000 PCI 00001 PCI Express 2 00010 PCI Express 1 00011 PCI Express 3 00111 Security 01010 Boot sequencer 10000 Processor instruction 10001 Processor data 10101 DMA 11000 eTSEC1 11010 eTSEC3 All other settings reserved.
31	ECV	Error capture valid. This bit indicates that the capture registers 0-3 contain valid info. This bit when set indicates that the captured registers contain valid capturing information. No new capturing will be done unless this bit is cleared by writing a 1 to it.

### 18.3.6.5 PCI Express Error Capture Register 0 (PEX\_ERR\_CAP\_R0)

Together with the other PCI Express error capture registers, PEX\_ERR\_CAP\_R0 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX\_ERR\_CAP\_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX\_ERR\_CAP\_STAT[ECV] bit is clear.

PEX\_ERR\_CAP\_R0 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX\_ERR\_CAP\_STAT[GSID] ≠ 0h02), is shown in Figure 18-28.



**Figure 18-28. PCI Express Error Capture Register 0 (PEX\_ERR\_CAP\_R0) Internal Source, Outbound Transaction**

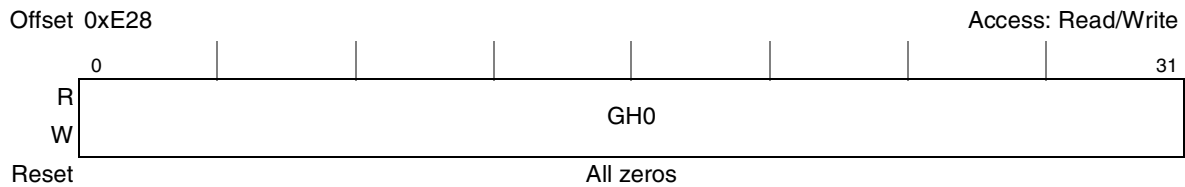


Table 18-27 describes the fields of the PCI Express error capture register 0 for the case when the error is caused by an outbound transaction from an internal source.

**Table 18-27. PCI Express Error Capture Register 0 Field Descriptions  
Internal Source, Outbound Transaction**

Bits	Name	Description
0–15	—	Reserved
16–17	FMT	PCI Express format. This field indicates the PCI Express packet format. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
18–22	TYPE	PCI Express type. This field indicates the PCI express packet type. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
23–31	—	Reserved

PEX\_ERR\_CAP\_R0 for the case when the error is caused by an inbound transaction from an external source (that is, PEX\_ERR\_CAP\_STAT[GSID] = 0h02 for controller 1), is shown in Figure 18-29.



**Figure 18-29. PCI Express Error Capture Register 0 (PEX\_ERR\_CAP\_R0)  
External Source, Inbound Transaction**

Table 18-28 describes the fields of PEX\_ERR\_CAP\_R0 for the case when the error is caused by an inbound transaction from an external source.

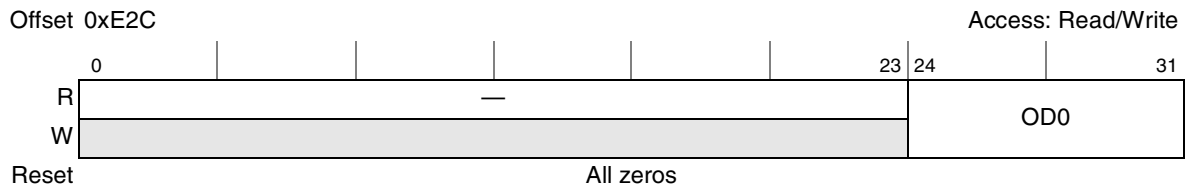
**Table 18-28. PCI Express Error Capture Register 0 Field Descriptions  
External Source, Inbound Transaction**

Bits	Name	Description
0–31	GH0	PCI Express 1st DW (4-byte) header. This field contains the PCI Express error packet's 1st DW (4-byte) header. 27–31 type 26–26 fmt 20–24 Rsv 17–19 TC 16 Rsv 14–15 length[9:8] 12–13 Rsv 10–11 Attr 9 EP 8 TD 0–7 length[7:0]

### 18.3.6.6 PCI Express Error Capture Register 1 (PEX\_ERR\_CAP\_R1)

Together with the other PCI Express error capture registers, PEX\_ERR\_CAP\_R1 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX\_ERR\_CAP\_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX\_ERR\_CAP\_STAT[ECV] bit is clear.

PEX\_ERR\_CAP\_R1 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX\_ERR\_CAP\_STAT[GSID] ≠ 0h02), is shown in Figure 18-30.



**Figure 18-30. PCI Express Error Capture Register 1 (PEX\_ERR\_CAP\_R1)  
Internal Source, Outbound Transaction**

Table 18-29 describes the fields of PEX\_ERR\_CAP\_R1 for the case when the error is caused by an outbound transaction from an internal source.

**Table 18-29. PCI Express Error Capture Register 1 Field Descriptions  
Internal Source, Outbound Transaction**

Bits	Name	Description
0–23	—	Reserved
24–31	OD0	Internal platform header bits 71-64. Contains internal platform header bit 71:64. Note that this field is only valid for outbound internal platform errors. It is not valid for completion time-out errors or PEX_CONFIG_ADDR/PEX_CONFIG_DATA errors. 30–31 Cls 26–29 Tag for read and non-posted write. Reserved otherwise. 25 Reserved 24 With Response (1 for read and non-posted write)

PEX\_ERR\_CAP\_R1 for the case when the error is caused by an inbound transaction from an external source (that is, PEX\_ERR\_CAP\_STAT[GSID] = 0h02 for controller 1), is shown in Figure 18-31.



**Figure 18-31. PCI Express Error Capture Register 1 (PEX\_ERR\_CAP\_R1)  
External Source, Inbound Transaction**

Table 18-30 describes the fields of PEX\_ERR\_CAP\_R1 for the case when the error is caused by an inbound transaction from an external source.

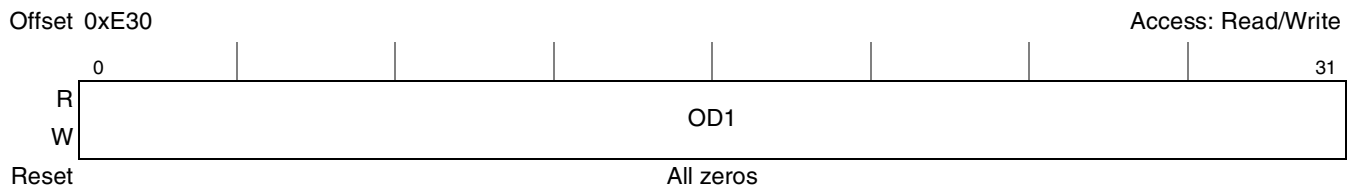
**Table 18-30. PCI Express Error Capture Register 1 Field Descriptions  
External Source, Inbound Transaction**

Bits	Name	Description
0–31	GH1	PEX 2nd DW (4-byte) header. This field contains the PCI Express error packet's 2nd DW (4-byte) header.
24–31		Comp ID[15:8]
16–23		Comp ID[7:0]
13–15		Comp Status
12		BCM
8–11		Byte Count[11:8]
0–7		Byte Count[7:0]

### 18.3.6.7 PCI Express Error Capture Register 2 (PEX\_ERR\_CAP\_R2)

Together with the other PCI Express error capture registers, PEX\_ERR\_CAP\_R2 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX\_ERR\_CAP\_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX\_ERR\_CAP\_STAT[ECV] bit is clear.

PEX\_ERR\_CAP\_R2 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX\_ERR\_CAP\_STAT[GSID] ≠ 0h02), is shown in Figure 18-32.



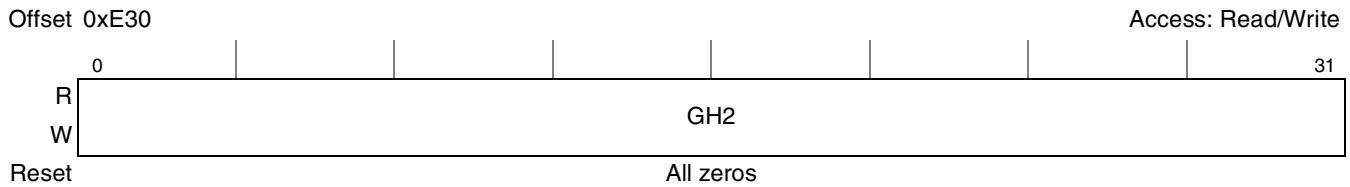
**Figure 18-32. PCI Express Error Capture Register 2 (PEX\_ERR\_CAP\_R2)  
Internal Source, Outbound Transaction**

Table 18-29 describes the fields of PEX\_ERR\_CAP\_R2 for the case when the error is caused by an outbound transaction from an internal source.

**Table 18-31. PCI Express Error Capture Register 2 Field Descriptions  
Internal Source, Outbound Transaction**

Bit	Name	Description
0–31	OD1	Internal platform header bits 63-32. Contains the internal platform header bits 63–32. Note that this field is only valid for outbound internal platform errors. It is not valid for completion time-out errors or PEX_CONFIG_ADDR/PEX_CONFIG_DATA errors. 27–31 Transaction type (00000 for write, 00001 for read) 25–26 Transaction priority 19-24 Source ID 10-18 Size 9 No snoop 8 Stash 7 Lock 1-6 Reserved 0 Address[0]

PEX\_ERR\_CAP\_R2 for the case when the error is caused by an inbound transaction from an external source (that is, PEX\_ERR\_CAP\_STAT[GSID] = 0h02 for controller 1), is shown in Figure 18-31.



**Figure 18-33. PCI Express Error Capture Register 2 (PEX\_ERR\_CAP\_R2)  
External Source, Inbound Transaction**

Table 18-30 describes the fields of PEX\_ERR\_CAP\_R2 for the case when the error is caused by an inbound transaction from an external source.

**Table 18-32. PCI Express Error Capture Register 2 Field Descriptions  
External Source, Inbound Transaction**

Bits	Name	Description
0–31	GH2	PCI Express 3rd DW (4-byte) header. This field contains the PCI Express error packet's 3rd DW (4-byte) header. 24–31 Req ID[15:8] 16–23 Req ID[7:0] 8–15 Tag[7:0] 1–7 Lower Address[6:0] 0 Rsv

### 18.3.6.8 PCI Express Error Capture Register 3 (PEX\_ERR\_CAP\_R3)

Together with the other PCI Express error capture registers, PEX\_ERR\_CAP\_R3 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound

transaction from an external source; the source of the captured error is reflected in PEX\_ERR\_CAP\_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX\_ERR\_CAP\_STAT[ECV] bit is clear.

PEX\_ERR\_CAP\_R3 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX\_ERR\_CAP\_STAT[GSID] ≠ 0h02), is shown in Figure 18-34.



**Figure 18-34. PCI Express Error Capture Register 3 (PEX\_ERR\_CAP\_R3)  
Internal Source, Outbound Transaction**

Table 18-29 describes the fields of PEX\_ERR\_CAP\_R3 for the case when the error is caused by an outbound transaction from an internal source.

**Table 18-33. PCI Express Error Capture Register 3 Field Descriptions  
Internal Source, Outbound Transaction**

Bits	Name	Description
0–31	OD2	Internal platform header bits 31-0. Contains internal platform header bits 31-0. Note that this field is only valid for outbound internal platform errors. It is not valid for completion time-out errors or PEX_CONFIG_ADDR/PEX_CONFIG_DATA errors. 0–31Address[1:32]

PEX\_ERR\_CAP\_R3 for the case when the error is caused by an inbound transaction from an external source (that is, PEX\_ERR\_CAP\_STAT[GSID] = 0h02 for controller 1), is shown in Figure 18-31.



**Figure 18-35. PCI Express Error Capture Register 3 (PEX\_ERR\_CAP\_R3)  
External Source, Inbound Transaction**

Table 18-30 describes the fields of PEX\_ERR\_CAP\_R3 for the case when the error is caused by an inbound transaction from an external source.

**Table 18-34. PEX Error Capture Register 3 Field Descriptions  
External Source, Inbound Transaction**

Bits	Name	Description
0–31	GH3	PEX 4th DW (4-byte) header. This field is a don't care.

## 18.3.7 PCI Express Configuration Space Access

There are two methods of accessing the PCI Express configuration header:

- PCI Express outbound ATMU window
- PCI Express configuration access registers (PEX\_CONFIG\_ADDR/PEX\_CONFIG\_DATA)

### 18.3.7.1 RC Configuration Register Access

To access internal configuration space, software must rely on the PCI Express configuration access register (PEX\_CONFIG\_ADDR/ PEX\_CONFIG\_DATA) mechanism. To access external configuration space, software can either use configuration access registers or the outbound ATMU mechanism. For the configuration access register method, a value must be written to the PEX\_CONFIG\_ADDR register that specifies the PCI Express bus, the device on that bus, the function within the device, and the configuration register in that device that should be accessed. The PCI Express controller's bus number is obtained from the PCI Express configuration header (type 1). Then either a write or a read to the PEX\_CONFIG\_DATA register triggers the actual write or read cycle to the configuration space. Note that accesses to the little-endian PCI Express configuration space must be properly formatted. See [Section 18.4.1.2.1, "Byte Order for Configuration Transactions,"](#) for more information.

Note that external configuration transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX\_LTSSM\_STAT) to check the status of link training before issuing external configuration requests.

#### 18.3.7.1.1 PCI Express Configuration Access Register Mechanism

There are two types of configuration transactions (Type 0 and Type 1) needed to support hierarchical bridges.

- If the bus number, and device number equal to the PCI Express controller's bus number and device number, and the function number is zero, then an internal PCI Express configuration cycle access is performed.
- If the bus number does not equal the PCI Express controller's bus number, but does equal the secondary bus number (from the type 1 header) and the device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the bus number does not equal the PCI Express controller's bus number, and does not equal the secondary bus number (from the type 1 header), and the bus number is less than or equal to the subordinate bus number (from the type 1 header), then a Type 1 configuration transaction is sent to the PCI Express link.
- If none of the above conditions occur, then the PCI Express controller returns all 1s for reads and ignores writes.

#### 18.3.7.1.2 Outbound ATMU Configuration Mechanism (RC-Only)

Software can also program one of the outbound ATMU windows to perform a configuration access. This is accomplished by programming the ReadTType or WriteTType field of the desired PEXOWAR to 0x2. Software must only issue 4-byte or less access to the ATMU configuration window and the access cannot

cross a 4-byte boundary. The bus number, device number, function number, register, and extended register number sent are decoded from the outbound translated PCI Express address.

- bus number[7:0] = PCI Express address[27:20]
- device number[4:0] = PCI Express address[19:15]
- function number[2:0] = PCI Express address[14:12]
- extended register number[3:0] = PCI Express address[11:8]
- register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the bus number equals the secondary bus number (from the type 1 header) and device number is 0. A Type 1 configuration cycle is sent to the link if bus number does not equal primary bus and secondary bus numbers and it is less than or equal to the subordinate bus number (from the type 1 header). For all other cases, the PCI Express controller squashes the write and read will result in a response with error returned.

Note that the PCI Express controller does not support access to its internal configuration registers using the outbound ATMU mechanism. That is, the outbound ATMU mechanism must not be used to program the internal registers.

### 18.3.7.2 EP Configuration Register Access

When the PCI Express controller is configured as an EP device it responds to remote host generated configuration cycles. This is indicated by decoding the configuration command along with type 0 access in the packet. A remote host can access up to 4096 bytes of the PCI Express configuration area. While in EP mode, the PCI Express controller does not support generating configuration accesses as a master. All accesses to PEX\_CONFIG\_ADDR/PEX\_CONFIG\_DATA cause the device to access the internal configuration registers regardless of the bus number or device number programmed in the PEX\_CONFIG\_ADDR register. There is no configuration mechanism supported in EP mode using the ATMU window. If the outbound ATMU window is configured to issue a configuration transaction, all posted transactions hitting this window are ignored and all non-posted transactions will get a response with an error.

### 18.3.8 PCI Compatible Configuration Headers

The first 64 bytes of the 256-byte PCI compatible configuration space consists of a predefined header that every PCI device must support. The first 16 bytes of the predefined header are defined the same for all PCI Express devices. These common registers are shown in [Figure 18-36](#).

Reserved				Address Offset (Hex)
Device ID	Vendor ID			00
Status	Command			04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C

**Figure 18-36. PCI Express PCI-Compatible Configuration Header Common Registers**

The remaining 48 bytes of the header may have differing layouts depending on the function of the device. There are two header types applicable to PCI Express. Type 0 headers are typically used by endpoints; Type 1 headers are used by root complexes and switches/bridges.

### 18.3.8.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

#### 18.3.8.1.1 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in Figure 18-37, is used to identify the manufacturer of the device.

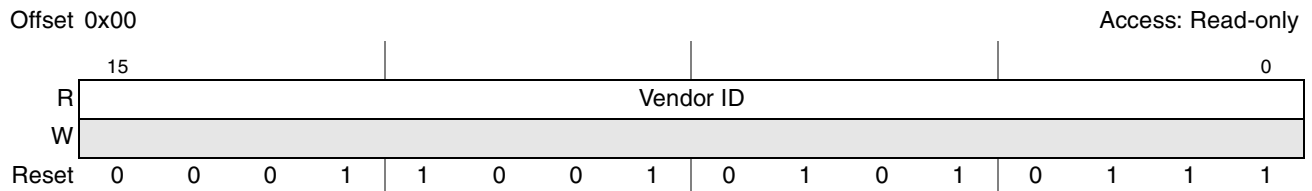


Figure 18-37. PCI Express Vendor ID Register

Table 18-35 describes the vendor ID register fields.

Table 18-35. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

#### 18.3.8.1.2 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in Figure 18-38, is used to identify the device.

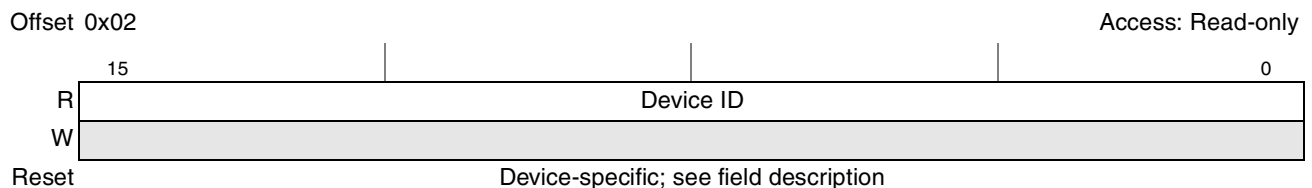


Figure 18-38. PCI Express Device ID Register

Table 18-36 describes the device ID register fields.

Table 18-36. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x0030 MPC8533E 0x0031 MPC8533

#### 18.3.8.1.3 PCI Express Command Register—Offset 0x04

The command register, shown in Figure 18-39, provides control over the ability to generate and respond to PCI Express cycles.



Offset 0x04

Access: Mixed

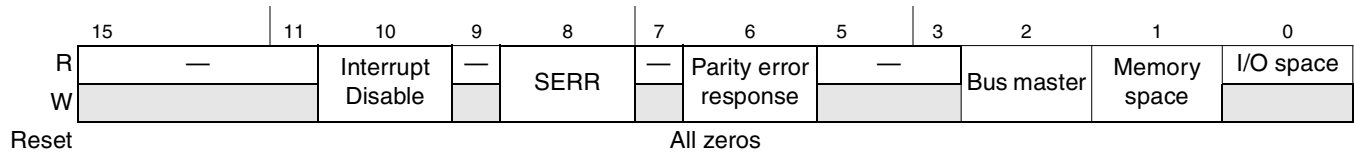


Figure 18-39. PCI Express Command Register

Table 18-37 describes the bits of the command register.

Table 18-37. PCI Express Command Register Field Descriptions

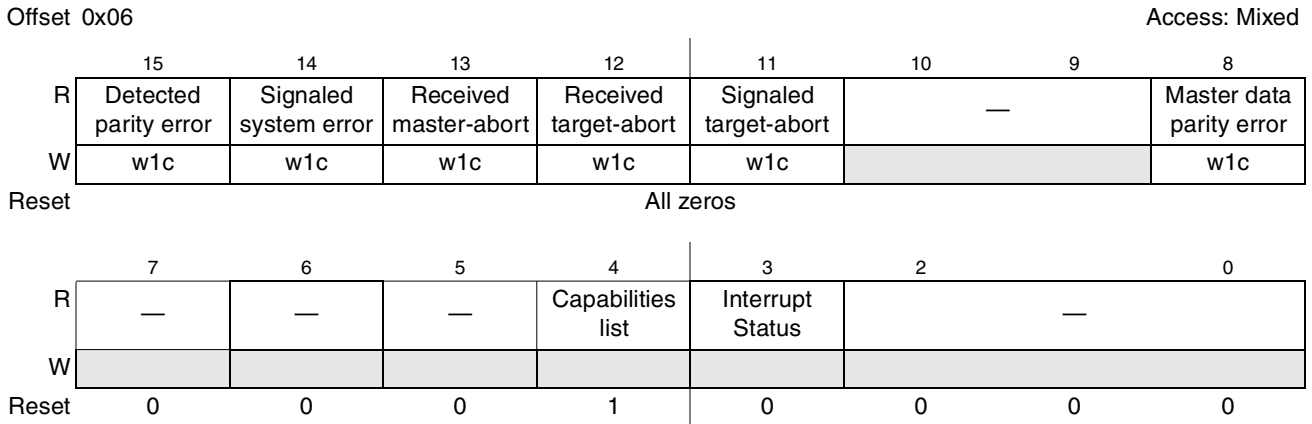
Bits	Name	Description
15–11	—	Reserved
10	Interrupt Disable	Controls the ability to generate INTx interrupt messages. 0 Enables INTx interrupt messages 1 Disables INTx interrupt messages Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set.
9	—	Reserved
8	SERR	Controls the reporting of fatal and non-fatal errors detected by the device to the root complex. 0 Disables reporting 1 Enables reporting <b>Note:</b> The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in <a href="#">Section 18.3.9.8, “PCI Express Device Control Register—0x54,”</a> and the advance error reporting capability structure described in sections 18.3.10.1 through 18.3.10.12.
7	—	Reserved
6	Parity error response	Controls whether this PCI Express controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Parity errors cause the appropriate bit in the PCI Express status register to be set. However, note that errors are reported based on the values set in the PCI Express error enable and detection registers. <b>Note:</b> The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in <a href="#">Section 18.3.9.8, “PCI Express Device Control Register—0x54,”</a> and the advance error reporting capability structure described in sections 18.3.10.1 through 18.3.10.12.
5–3	—	Reserved
2	Bus master	Indicates whether this PCI Express device is configured as a master. 0 Disables the ability to generate PCI Express accesses 1 Enables this PCI Express controller to behave as a PCI Express bus master EP mode: Clearing this bit prevent the device from issuing any memory or I/O transactions. Because MSI interrupts are effectively memory writes, clearing this bit also disables the ability of the device to issue MSI interrupts. RC mode: Clearing this bit disables the ability of the device to forward memory transactions upstream. This causes any inbound memory transaction to be treated as an unsupported request.

**Table 18-37. PCI Express Command Register Field Descriptions (continued)**

Bits	Name	Description
1	Memory space	Controls whether this PCI Express device (as a target) responds to memory accesses. 0 This PCI Express device does not respond to PCI Express memory space accesses. 1 This PCI Express device responds to PCI Express memory space accesses. EP mode: Clearing this bit will prevent the device from accepting any memory transaction. RC mode: This bit is ignored. It does not affect outbound memory transaction
0	I/O space	I/O space. 0 This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1 This PCI Express device (as a target) does respond to PCI Express I/O space accesses. EP mode: Clearing this bit will prevent the device from accepting any IO transaction. Note that this bit is a don't care in EP mode since the device does not support IO transaction. RC mode: This bit is ignored. It does not affect outbound IO transaction.

**18.3.8.1.4 PCI Express Status Register—Offset 0x06**

The status register, shown in [Figure 18-40](#), is used to record status information for PCI Express related events.



**Figure 18-40. PCI Express Status Register**

The definition of each bit is given in [Table 18-38](#).

**Table 18-38. PCI Express Status Register Field Descriptions**

Bits	Name	Description
15	Detected parity error <sup>1</sup>	Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register.
14	Signaled system error <sup>1</sup>	Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.
13	Received master-abort <sup>1</sup>	Set whenever a requestor receives a completion with unsupported request completion status.
12	Received target-abort <sup>1</sup>	Set whenever a device receives a completion with completer abort completion status.
11	Signaled target-abort <sup>1</sup>	Set whenever a device completes a request using completer abort completion status.

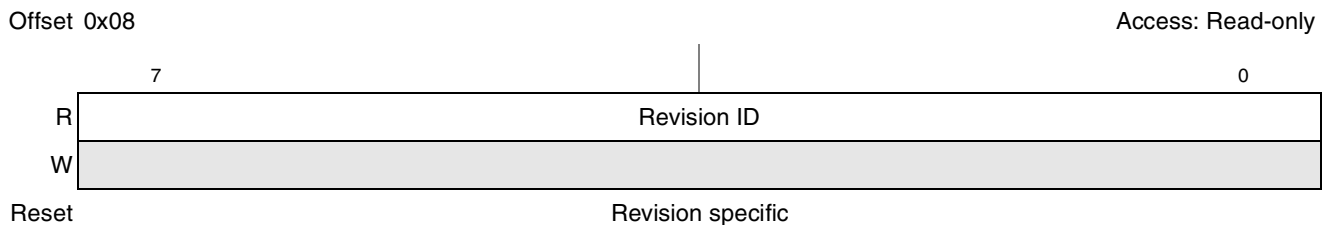
**Table 18-38. PCI Express Status Register Field Descriptions (continued)**

Bits	Name	Description
10–9	—	Reserved
8	Master data parity error detected <sup>1</sup>	Set by the requestor (primary side for Type1 headers) when either the requestor receives a completion marked poisoned or the requestor poisons a write request. Note that the parity error enable bit (bit 6) in the command register must be set for this bit to be set.
7–5	—	Reserved
4	Capabilities List	All PCI Express devices are required to implement the PCI Express capability structure.
3	Interrupt Status	Set when an INTx interrupt message is pending internally to the device. Note that this bit is associated with INTx messages and not Message Signaled Interrupts.
2–0	—	Reserved

<sup>1</sup> The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in [Section 18.3.9.8, “PCI Express Device Control Register—0x54,”](#) and the advance error reporting capability structure described in sections 18.3.10.1 through 18.3.10.12.

### 18.3.8.1.5 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in [Figure 18-41](#), is used to identify the revision of the device.

**Figure 18-41. PCI Express Revision ID Register**

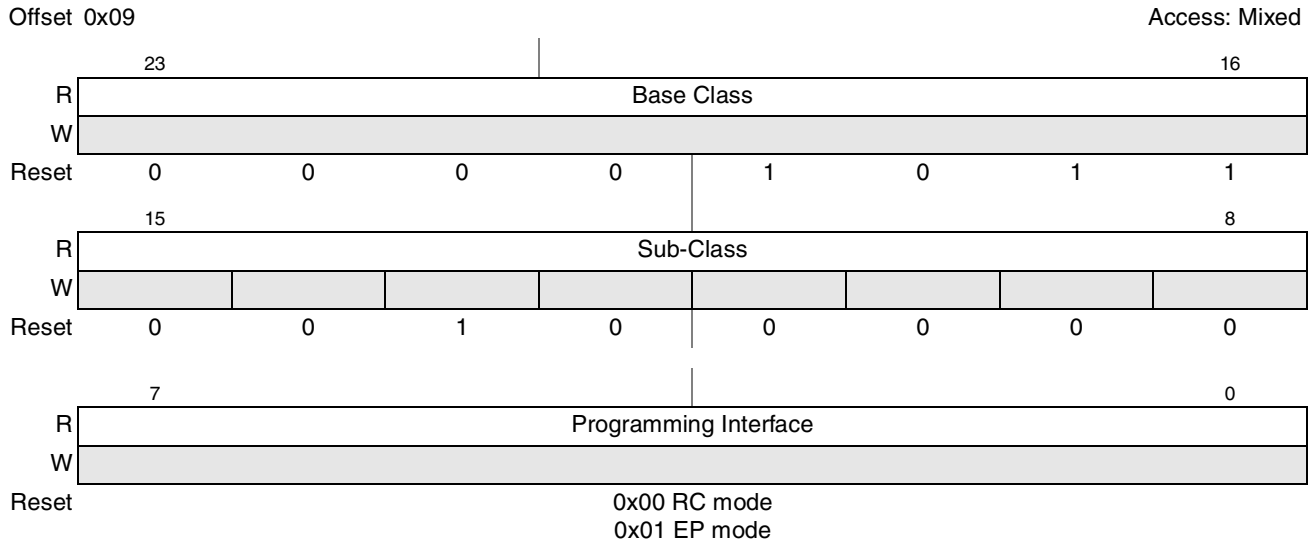
[Table 18-39](#) describes the revision ID register fields.

**Table 18-39. PCI Express Revision ID Register Field Descriptions**

Bits	Name	Description
7–0	Revision ID	Revision specific.

### 18.3.8.1.6 PCI Express Class Code Register—Offset 0x09

The class code register, shown in [Figure 18-42](#), is comprised of three single-byte fields—base class (offset 0x0B), sub-class (offset 0x0A), and programming interface (offset 0x09)—that indicate the basic functionality of the function.



**Figure 18-42. PCI Express Class Code Register**

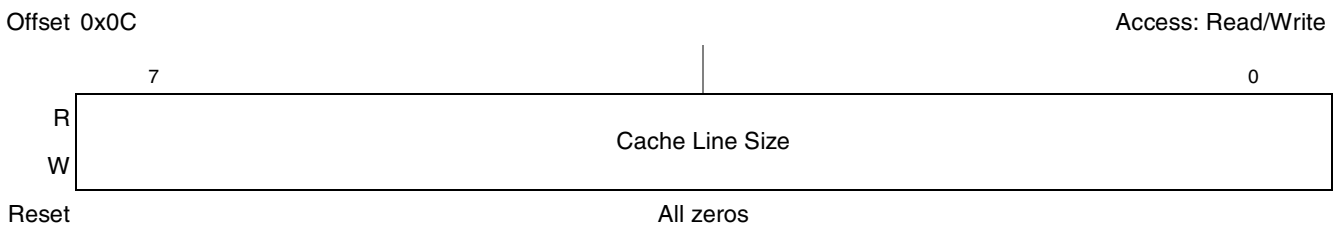
Table 18-40 describes the class code register fields.

**Table 18-40. PCI Express Class Code Register Field Descriptions**

Bits	Name	Description
23–16	Base Class	0x0B—Processor
15–8	Sub-Class	0x20—PowerPC
7–0	Programming Interface	0x00—RC mode 0x01—EP mode

### 18.3.8.1.7 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in Figure 18-43, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.



**Figure 18-43. PCI Express Bus Cache Line Size Register**

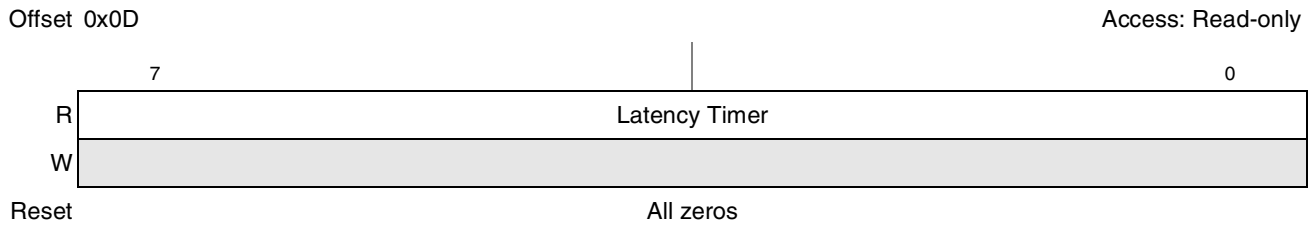
Table 18-41 describes the cache line size register.

**Table 18-41. PCI Express Bus Cache Line Size Register Field Descriptions**

Bits	Name	Description
7–0	Cache Line Size	Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). Note that for PCI Express operation this register is ignored.

### 18.3.8.1.8 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in [Figure 18-44](#), is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.



**Figure 18-44. PCI Express Bus Latency Timer Register**

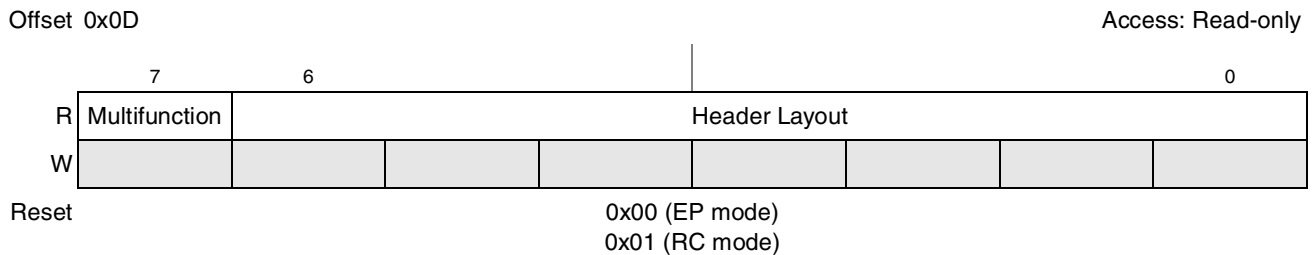
[Table 18-42](#) describes the PCI Express latency timer register (PLTR).

**Table 18-42. PCI Express Bus Latency Timer Register Field Descriptions**

Bits	Name	Description
7–0	Latency Timer	Note that for PCI Express operation this register is ignored.

### 18.3.8.1.9 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in [Figure 18-43](#), is used to identify the layout of the PCI compatible header.



**Figure 18-45. PCI Express Bus Latency Timer Register**

[Table 18-42](#) describes the PCI Express header type register.

**Table 18-43. PCI Express Bus Latency Timer Register Field Descriptions**

Bits	Name	Description
7	Multifunction	Identifies whether a device supports multiple functions 0 Single function device 1 Multiple function device
6–0	Header Layout	0x00 Endpoint. See <a href="#">Figure 18-46</a> for type 0 layout. 0x01 Root Complex. See <a href="#">Figure 18-58</a> for type 1 layout. All other encodings reserved.

### 18.3.8.1.10 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

### 18.3.8.2 Type 0 Configuration Header

The type 0 header is shown in [Figure 18-46](#).

				Address Offset (Hex)	
Device ID				Vendor ID	00
Status		Command		04	
Class Code			Revision ID	08	
BIST	Header Type	Latency Timer	Cache Line Size	0C	
Base Address Registers				10	
				14	
				18	
				1C	
				20	
Base Address Registers				24	
Base Address Registers				28	
Subsystem ID		Subsystem Vendor ID		2C	
Base Address Registers				30	
Base Address Registers			Capabilities Pointer	34	
Expansion ROM Base Address				38	
MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line	3C	

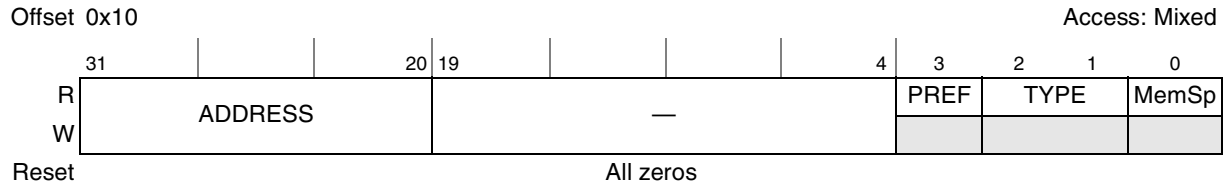
**Figure 18-46. PCI Express PCI-Compatible Configuration Header—Type 0**

[Section 18.3.8.1, “Common PCI Compatible Configuration Header Registers,”](#) describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 0 header beginning at offset 0x10.

#### 18.3.8.2.1 PCI Express Base Address Registers—0x10–0x27

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to [Section 18.3.5.2, “PCI Express Inbound ATMU Registers,”](#) for more information.

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in [Figure 18-47](#).



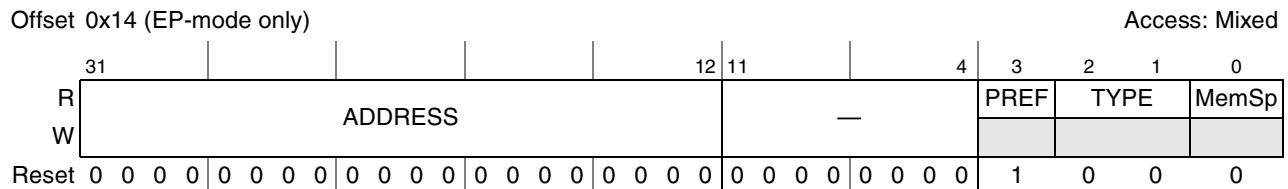
**Figure 18-47. PCI Express Base Address Register 0 (PEXCSRBAR)**

Table 18-44 describes the PCI Express configuration and status register base address register.

**Table 18-44. PEXCSRBAR Field Descriptions**

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space. The 32-bit memory BAR is shown in Figure 18-48.



**Figure 18-48. 32-Bit Memory Base Address Register (BAR1)**

Table 18-45 describes the PCI Express 32-bit memory BAR fields.

**Table 18-45. 32-Bit Memory Base Address Register (BAR1) Field Descriptions**

Bits	Name	Description
31–12	ADDRESS	Indicates the base address where the inbound memory window begins. The number of upper bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes register (PEXIWAR1).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXIWAR1[PF].
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator.

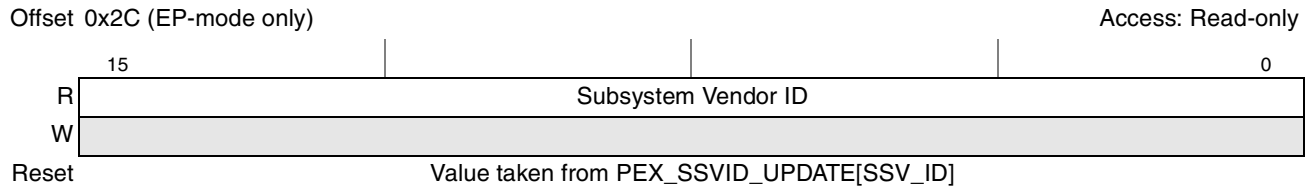
Base address register 2 at offset 0x18 and base address register 4 at offset 0x20 are used to define the lower portion of the 64-bit inbound memory windows. The 64-bit low memory BARs are shown in Figure 18-49.





### 18.3.8.2.2 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.



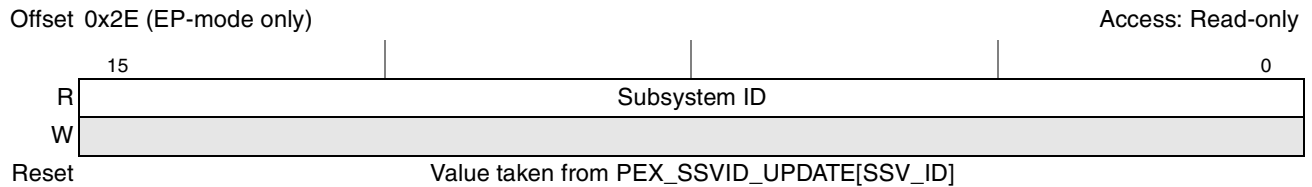
**Figure 18-51. PCI Express Subsystem Vendor ID Register**

**Table 18-48. PCI Express Subsystem Vendor ID Register Field Description**

Bits	Name	Description
15–0	Subsystem Vendor ID	The value for subsystem vendor ID is determined by the PCI Express subsystem vendor ID update register. See <a href="#">Section 18.3.10.17, “PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,”</a> for more information.

### 18.3.8.2.3 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.



**Figure 18-52. PCI Express Subsystem ID Register**

**Table 18-49. PCI Express Subsystem ID Register Field Description**

Bits	Name	Description
15–0	Subsystem ID	The value for subsystem ID is determined by the PCI Express subsystem vendor ID update register. See <a href="#">Section 18.3.10.17, “PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,”</a> for more information.

### 18.3.8.2.4 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

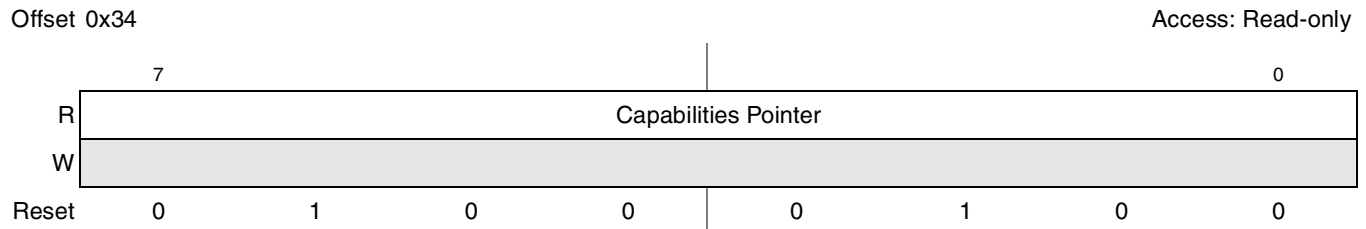


Figure 18-53. Capabilities Pointer Register

Table 18-50. Capabilities Pointer Register Field Description

Bits	Name	Description
7-0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to <a href="#">Section 18.3.9, “PCI Compatible Device-Specific Configuration Space,”</a> for more information.

### 18.3.8.2.5 PCI Express Interrupt Line Register (EP-Mode Only)—0x3C

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

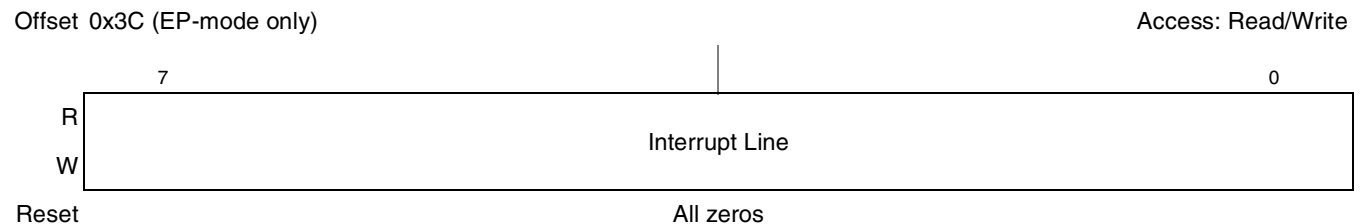


Figure 18-54. PCI Express Interrupt Line Register

Table 18-51. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7-0	Interrupt Line	Used to communicate interrupt line routing information.

### 18.3.8.2.6 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

Offset 0x3D

Access: Read-only

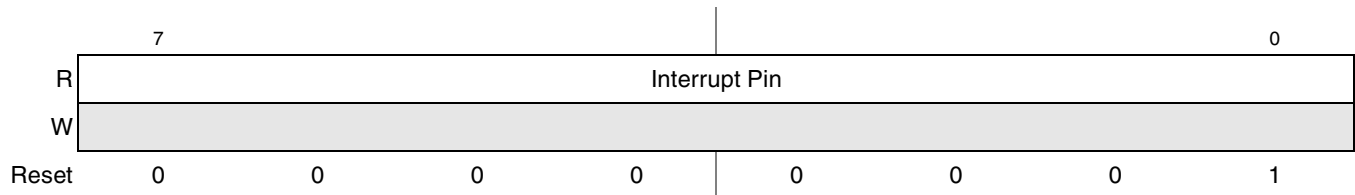


Figure 18-55. PCI Express Interrupt Pin Register

Table 18-52. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

### 18.3.8.2.7 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

Offset 0x3E (EP-mode only)

Access: Read-only

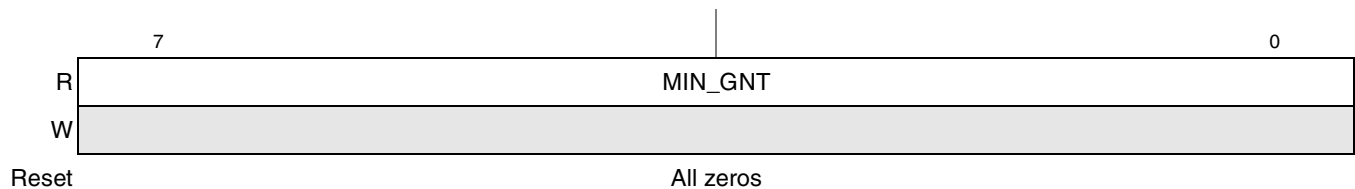


Figure 18-56. PCI Express Maximum Grant Register (MAX\_GNT)

Table 18-53. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

### 18.3.8.2.8 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

Offset 0x3F (EP-mode only)

Access: Read-only

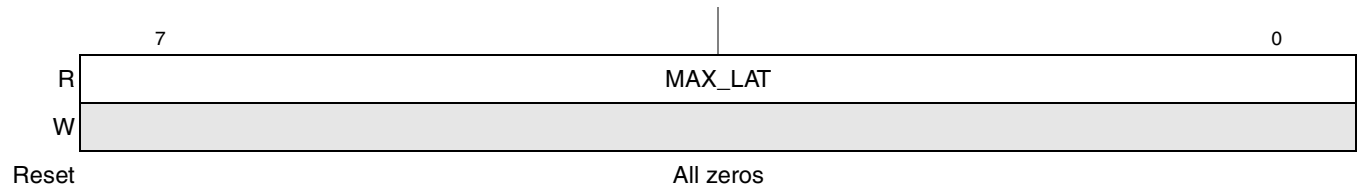


Figure 18-57. PCI Express Maximum Latency Register (MAX\_LAT)

Table 18-54. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7-0	MAX_LAT	Does not apply for PCI Express.

### 18.3.8.3 Type 1 Configuration Header

The type 1 header is shown in [Figure 18-58](#).

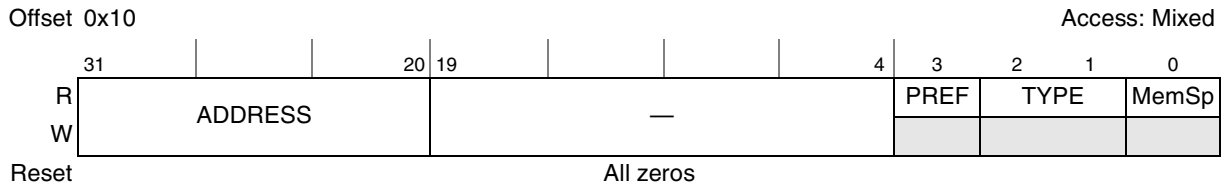
				Address Offset (Hex)
Reserved				
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Register 0				10
Reserved				14
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18
Secondary Status		I/O Limit	I/O Base	1C
Memory Limit		Memory Base		20
Prefetchable Memory Limit		Prefetchable Memory Base		24
Prefetchable Base Upper 32 Bits				28
Prefetchable Limit Upper 32 Bits				2C
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30
Reserved			Capabilities Pointer	34
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 18-58. PCI Express PCI-Compatible Configuration Header—Type 1

Section 18.3.8.1, “Common PCI Compatible Configuration Header Registers,” describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 1 header beginning at offset 0x10.

### 18.3.8.3.1 PCI Express Base Address Register 0—0x10

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in [Figure 18-47](#).



**Figure 18-59. PCI Express Base Address Register 0 (PEXCSRBAR)**

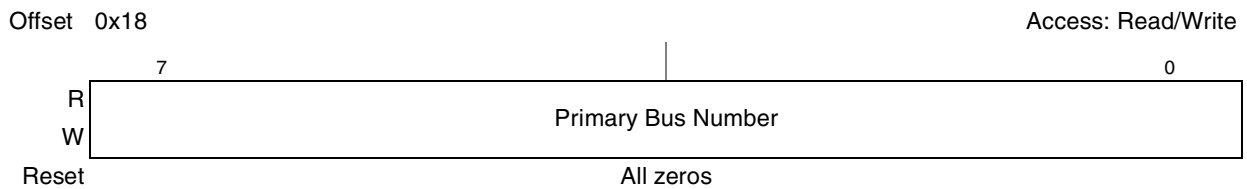
[Table 18-44](#) describes the PCI Express configuration and status register base address register.

**Table 18-55. PEXCSRBAR Field Descriptions**

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PEF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

### 18.3.8.3.2 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in [Figure 18-60](#).



**Figure 18-60. PCI Express Primary Bus Number Register**

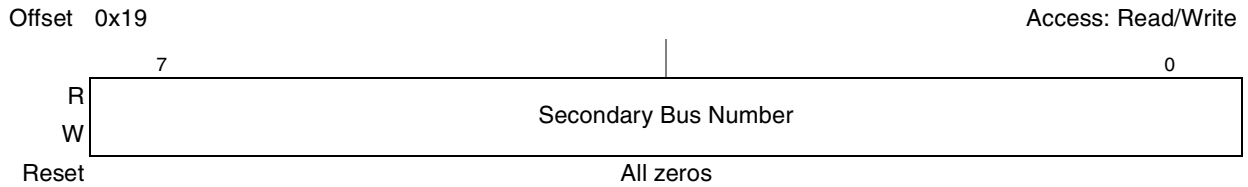
[Table 18-56](#) describes the primary bus number register fields.

**Table 18-56. PCI Express Primary Bus Number Register Field Description**

Bits	Name	Description
7–0	Primary Bus Number	Bus that is connected to the upstream interface. Note that this register is programmed during system enumeration; in RC mode this register should remain 0x00.

### 18.3.8.3.3 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in [Figure 18-61](#).



**Figure 18-61. PCI Express Secondary Bus Number Register**

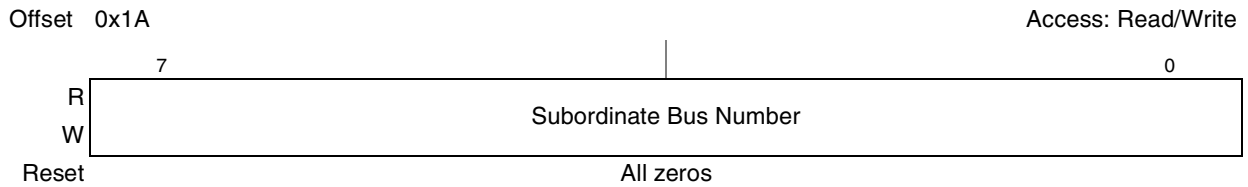
[Table 18-57](#) describes the secondary bus number register fields.

**Table 18-57. PCI Express Secondary Bus Number Register Field Description**

Bits	Name	Description
7–0	Secondary Bus Number	Bus that is directly connected to the downstream interface. Note that this register is programmed during system enumeration; in RC mode, this register is typically programmed to 0x01.

### 18.3.8.3.4 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in [Figure 18-62](#).



**Figure 18-62. PCI Express Subordinate Bus Number Register**

[Table 18-58](#) describes the subordinate bus number register fields.

**Table 18-58. PCI Express Subordinate Bus Number Register Field Description**

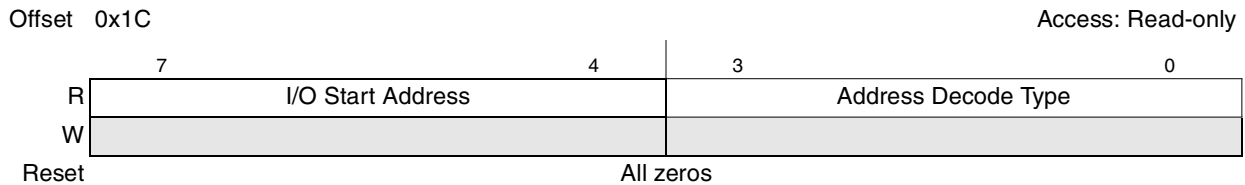
Bits	Name	Description
7–0	Subordinate Bus Number	Highest bus number that is on the downstream interface.

### 18.3.8.3.5 PCI Express Secondary Latency Timer Register—0x1B

The secondary latency timer register does not apply to PCI Express. It must be read-only and return all zeros when read.

### 18.3.8.3.6 PCI Express I/O Base Register—0x1C

Note that this device does not support inbound I/O transactions. The I/O base register is shown in [Figure 18-62](#).



**Figure 18-63. PCI Express I/O Base Register**

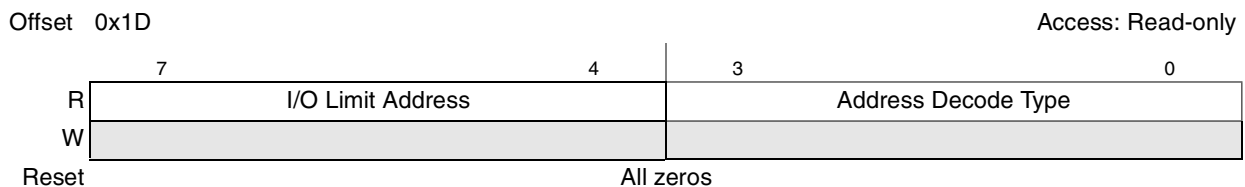
[Table 18-58](#) describes the I/O base register fields.

**Table 18-59. PCI Express I/O Base Register Field Description**

Bits	Name	Description
7–4	I/O Start Address	Specifies bits 15:12 of the I/O space start address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

### 18.3.8.3.7 PCI Express I/O Limit Register—0x1D

Note that this device does not support inbound I/O transactions. The I/O limit register is shown in [Figure 18-62](#).



**Figure 18-64. PCI Express I/O Limit Register**

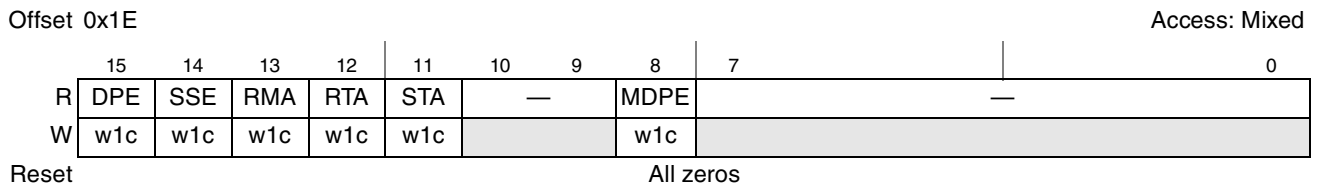
Table 18-58 describes the I/O limit register fields.

**Table 18-60. PCI Express I/O Limit Register Field Description**

Bits	Name	Description
7–4	I/O Limit Address	Specifies bits 15:12 of the I/O space ending address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

### 18.3.8.3.8 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in Figure 18-65. Note that the errors in this register may be masked by corresponding bits in the secondary status interrupt mask register (PEX\_SS\_INTR\_MASK) and that by default all of the errors are masked. See Section 18.3.10.20, “Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0” for more information.



**Figure 18-65. PCI Express Secondary Status Register**

Table 18-61 describes the PCI Express secondary status register fields.

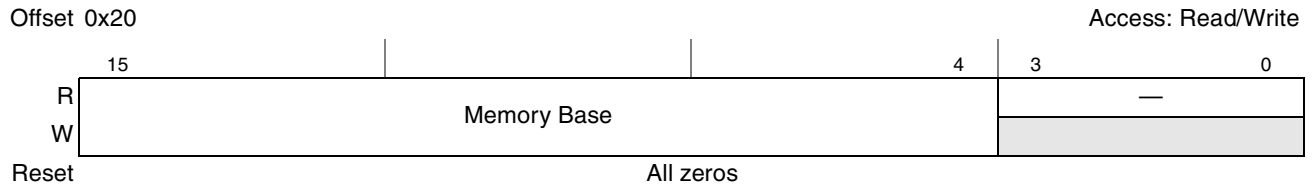
**Table 18-61. PCI Express Secondary Status Register Field Description**

Bits	Name	Description
15	DPE	Detected parity error. This bit is set whenever the secondary side receives a poisoned TLP regardless of the state of the parity error response bit.
14	SSE	Signaled system error. This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL message, provided the SERR enable bit in the command register is set to enable reporting.
13	RMA	Received master abort. This bit is set when the secondary side receives an unsupported request (UR) completion.
12	RTA	Received target abort. This bit is set when the secondary side receives a completer abort (CA) completion.
11	STA	Signaled target abort. This bit is set when the secondary side issues a CA completion.
10–9	—	Reserved
8	MDPE	Master data parity error. This bit is set when the parity error response bit is set and the secondary side requestor receives a poisoned completion or poisons a write request. If the parity error response bit is cleared, this bit is never set.
7–0	—	Reserved



### 18.3.8.3.9 PCI Express Memory Base Register—0x20

The memory base register is shown in [Figure 18-66](#).



**Figure 18-66. PCI Express Memory Base Register**

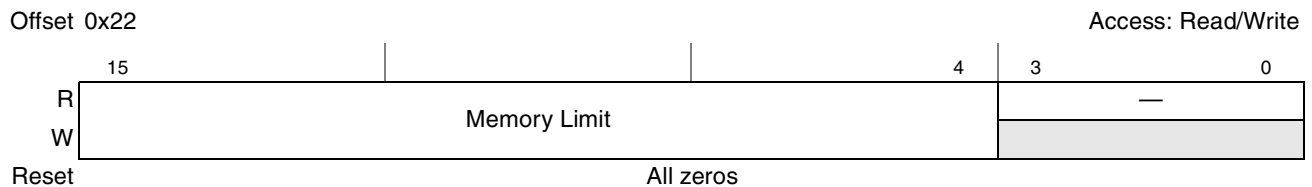
[Table 18-62](#) describes the memory base register fields.

**Table 18-62. PCI Express Memory Base Register Field Description**

Bits	Name	Description
15–4	Memory Base	Specifies bits 31:20 of the non-prefetchable memory space start address. Typically used for specifying memory-mapped I/O space. <b>Note:</b> Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in an unsupported request response.
3–0	—	Reserved

### 18.3.8.3.10 PCI Express Memory Limit Register—0x22

The memory limit register is shown in [Figure 18-67](#).



**Figure 18-67. PCI Express Memory Limit Register**

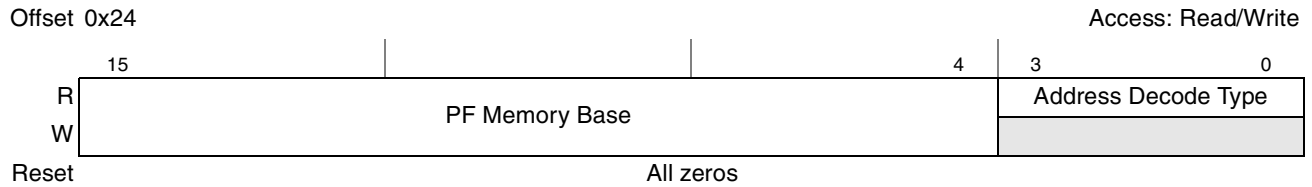
[Table 18-63](#) describes the memory base register fields.

**Table 18-63. PCI Express Memory Limit Register Field Description**

Bits	Name	Description
15–4	Memory Limit	Specifies bits 31:20 of the non-prefetchable memory space ending address. Typically used for specifying memory-mapped I/O space. <b>Note:</b> Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range will result in unsupported request response.
3–0	—	Reserved

### 18.3.8.3.11 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in [Figure 18-68](#).



**Figure 18-68. PCI Express Prefetchable Memory Base Register**

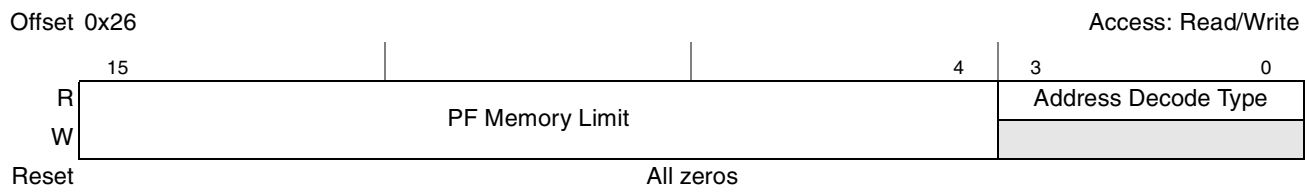
[Table 18-64](#) describes the prefetchable memory base register fields.

**Table 18-64. PCI Express Prefetchable Memory Base Register Field Description**

Bits	Name	Description
15–4	PF Memory Base	Specifies bits 31:20 of the prefetchable memory space start address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

### 18.3.8.3.12 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in [Figure 18-69](#).



**Figure 18-69. PCI Express Prefetchable Memory Limit Register**

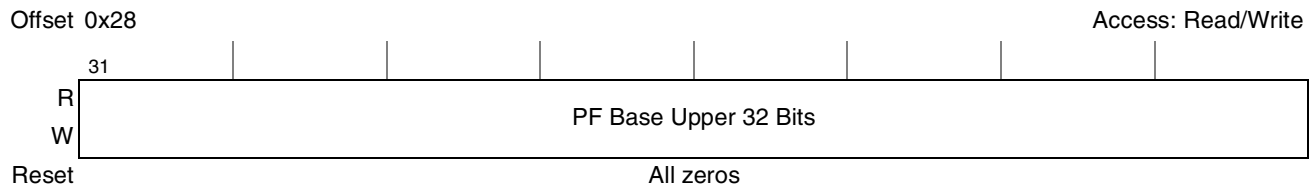
[Table 18-65](#) describes the prefetchable memory limit register fields.

**Table 18-65. PCI Express Prefetchable Memory Limit Register Field Description**

Bits	Name	Description
15–4	PF Memory Limit	Specifies bits 31:20 of the prefetchable memory space ending address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

### 18.3.8.3.13 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in [Figure 18-70](#).



**Figure 18-70. PCI Express Prefetchable Base Upper 32 Bits Register**

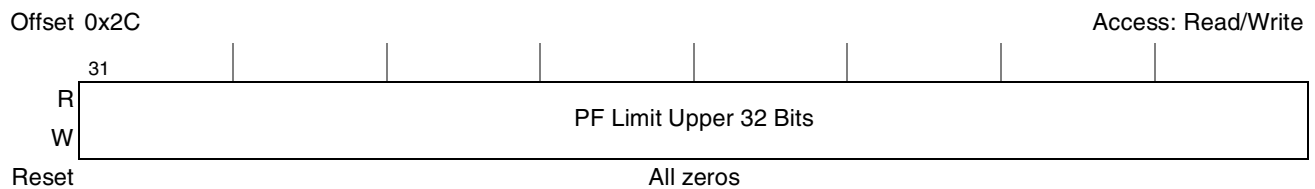
[Table 18-66](#) describes the PCI Express prefetchable memory base upper 32 bits register fields.

**Table 18-66. PCI Express Prefetchable Base Upper 32 Bits Register**

Bits	Name	Description
31–0	PF Base Upper 32 Bits	Specifies bits 64:32 of the prefetchable memory space start address when the address decode type field in the prefetchable memory base register is 0x01.

### 18.3.8.3.14 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory limit upper 32 bits register is shown in [Figure 18-71](#).



**Figure 18-71. PCI Express Prefetchable Limit Upper 32 Bits Register**

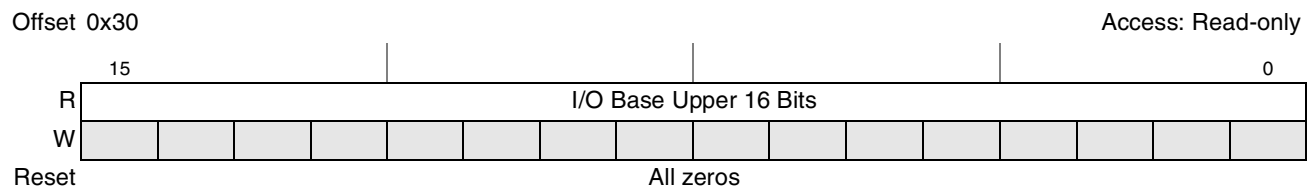
[Table 18-67](#) describes the PCI Express prefetchable memory limit upper 32 bits register fields.

**Table 18-67. PCI Express Prefetchable Limit Upper 32 Bits Register**

Bits	Name	Description
31–0	PF Limit Upper 32 Bits	Specifies bits 64–32 of the prefetchable memory space ending address when the address decode type field in the prefetchable memory limit register is 0x01.

### 18.3.8.3.15 PCI Express I/O Base Upper 16 Bits Register—0x30

Note that this device does not support inbound I/O transactions. The I/O base upper 16 bits register is shown in [Figure 18-72](#).



**Figure 18-72. PCI Express I/O Base Upper 16 Bits Register**

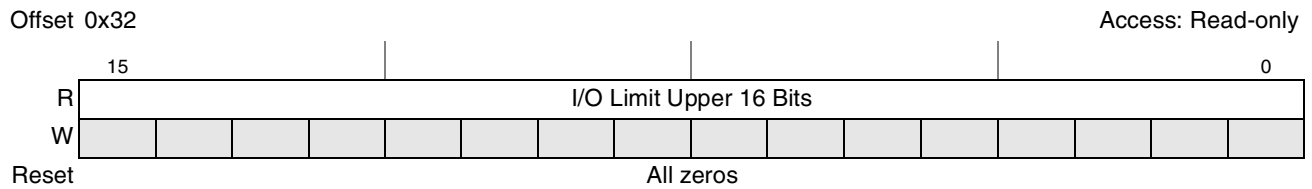
Table 18-68 describes the I/O base upper 16 bits register fields.

**Table 18-68. PCI Express I/O Base Upper 16 Bits Register Field Description**

Bits	Name	Description
15–0	I/O Base Upper 16 Bits	Specifies bits 31–16 of the I/O space start address when the address decode type field in the I/O base register is 0x01.

### 18.3.8.3.16 PCI Express I/O Limit Upper 16 Bits Register—0x32

Note that this device does not support inbound I/O transactions. The I/O limit upper 16 bits register is shown in Figure 18-73.



**Figure 18-73. PCI Express I/O Limit Upper 16 Bits Register**

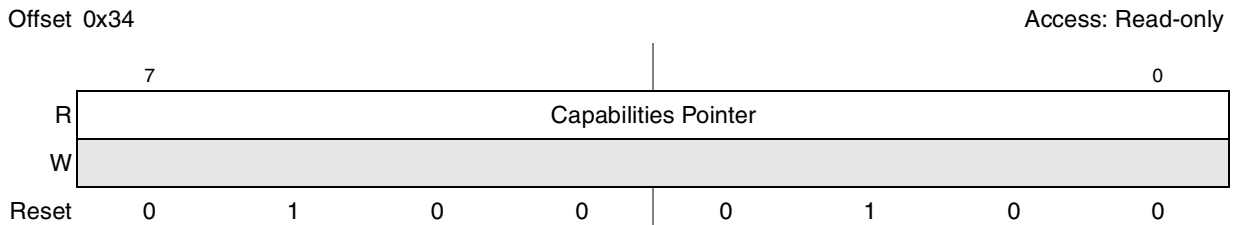
Table 18-69 describes the I/O limit upper 16 bits register fields.

**Table 18-69. PCI Express I/O Limit Upper 16 Bits Register Field Description**

Bits	Name	Description
15–0	I/O Limit Upper 16 Bits	Specifies bits 31–16 of the I/O space ending address when the address decode type field in the I/O limit register is 0x01.

### 18.3.8.3.17 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.



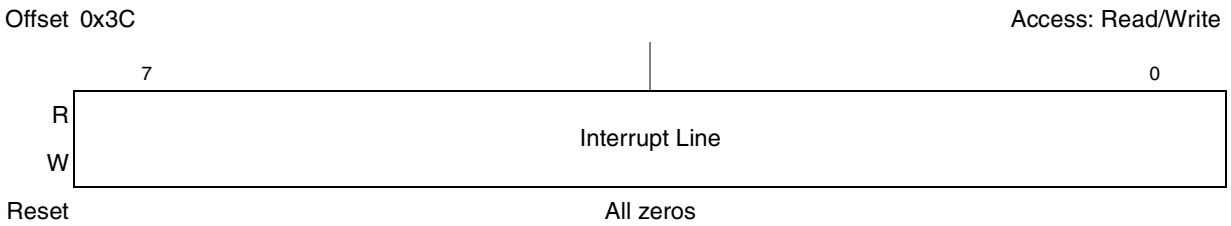
**Figure 18-74. Capabilities Pointer Register**

**Table 18-70. Capabilities Pointer Register Field Description**

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to Section 18.3.9, “PCI Compatible Device-Specific Configuration Space,” for more information.

### 18.3.8.3.18 PCI Express Interrupt Line Register—0x3C

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.



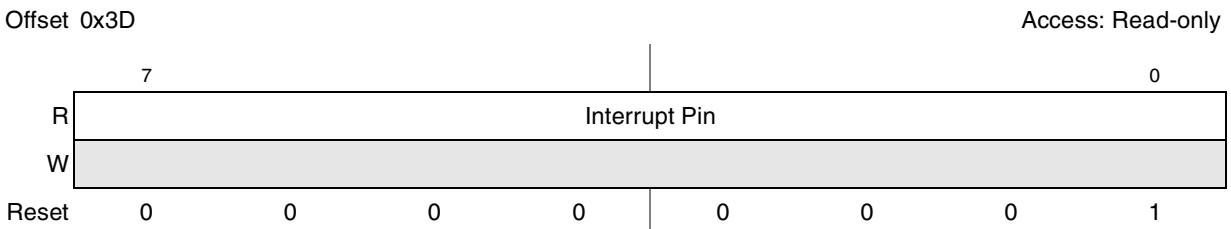
**Figure 18-75. PCI Express Interrupt Line Register**

**Table 18-71. PCI Express Interrupt Line Register Field Description**

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

### 18.3.8.3.19 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.



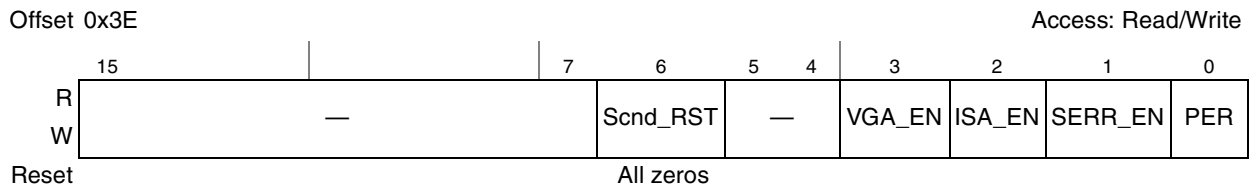
**Figure 18-76. PCI Express Interrupt Pin Register**

**Table 18-72. PCI Express Interrupt Pin Register Field Description**

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

### 18.3.8.3.20 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in [Figure 18-77](#).



**Figure 18-77. PCI Express Bridge Control Register**

[Table 18-73](#) describes the PCI Express bridge control register fields.

**Table 18-73. PCI Express Bridge Control Register Field Description**

Bits	Name	Description
15–7	—	Reserved
6	Scnd_RST	Secondary bus reset
5–4	—	Reserved
3	VGA_EN	VGA enable
2	ISA_EN	ISA enable
1	SERR_EN	SERR enable. This bit controls the propagation of ERR_COR, ERR_NONFATAL, and ERR_FATAL responses received on the secondary side.
0	PER	Parity error response.

### 18.3.9 PCI Compatible Device-Specific Configuration Space

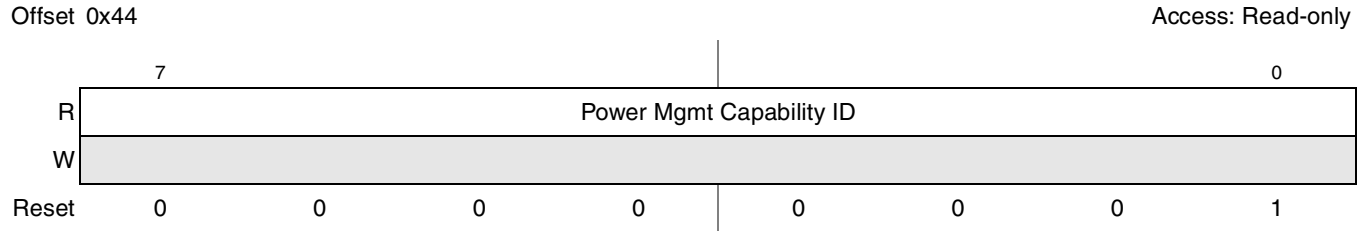
The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just above the 64-byte PCI-compatible configuration header).

Reserved	Address Offset (Hex)			
PCI-Compatible Configuration Header (See <a href="#">Section 18.3.8, "PCI Compatible Configuration Headers,"</a> for more information.)	00  3F			
	40			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Power Mgmt Capabilities</td> <td style="width: 33%;">Next Pointer (0x4C)</td> <td style="width: 33%;">Power Mgmt Capability ID</td> </tr> </table>	Power Mgmt Capabilities	Next Pointer (0x4C)	Power Mgmt Capability ID	44
Power Mgmt Capabilities	Next Pointer (0x4C)	Power Mgmt Capability ID		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">Data</td> <td style="width: 75%;">Power Management Status &amp; Control</td> </tr> </table>	Data	Power Management Status & Control	48	
Data	Power Management Status & Control			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 45%;">PCI Express Capabilities</td> <td style="width: 20%;">Next Pointer (0x70 — EP mode) (NULL — RC mode)</td> <td style="width: 35%;">PCI Express Capability ID</td> </tr> </table>	PCI Express Capabilities	Next Pointer (0x70 — EP mode) (NULL — RC mode)	PCI Express Capability ID	4C
PCI Express Capabilities	Next Pointer (0x70 — EP mode) (NULL — RC mode)	PCI Express Capability ID		
Device Capabilities	50			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Device Status</td> <td style="width: 50%;">Device Control</td> </tr> </table>	Device Status	Device Control	54	
Device Status	Device Control			
Link Capabilities	58			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Link Status</td> <td style="width: 50%;">Link Control</td> </tr> </table>	Link Status	Link Control	5C	
Link Status	Link Control			
Slot Capabilities	60			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Slot Status</td> <td style="width: 50%;">Slot Control</td> </tr> </table>	Slot Status	Slot Control	64	
Slot Status	Slot Control			
	68			
Root Status	6C			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">MSI Message Control</td> <td style="width: 33%;">Next Pointer (NULL)</td> <td style="width: 33%;">MSI Message Capability ID</td> </tr> </table>	MSI Message Control	Next Pointer (NULL)	MSI Message Capability ID	70
MSI Message Control	Next Pointer (NULL)	MSI Message Capability ID		
MSI Message Address	74			
MSI Upper Message Address	78			
	7C			
	80			
	FF			

Figure 18-78. PCI Compatible Device-Specific Configuration Space

### 18.3.9.1 PCI Express Power Management Capability ID Register—0x44

The PCI Express power management capability ID register is shown in [Figure 18-79](#).



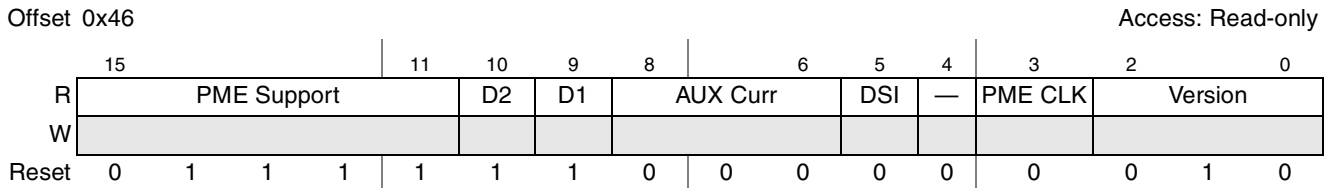
**Figure 18-79. PCI Express Power Management Capability ID Register**

**Table 18-74. PCI Express Power Management Capability ID Register Field Description**

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

### 18.3.9.2 PCI Express Power Management Capabilities Register—0x46

The PCI Express power management capabilities register is shown in [Figure 18-80](#).



**Figure 18-80. PCI Express Power Management Capabilities Register**

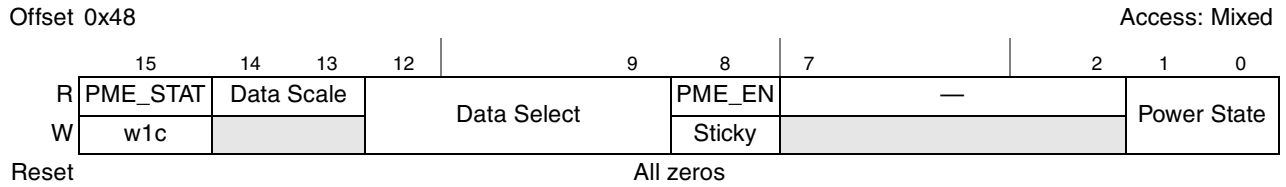
**Table 18-75. PCI Express Power Management Capabilities Register Field Description**

Bits	Name	Description
15–11	PME Support	Indicates the power states that this device supports
10	D2	D2 Support
9	D1	D1 Support
8–6	AUX Curr	AUX Current
5	DSI	Device Specific Initialization
4	—	Reserved
3	PME CLK	Does not apply to PCI Express.
2–0	Version	Version 1.0a



### 18.3.9.3 PCI Express Power Management Status and Control Register—0x48

The PCI Express power management status and control register is shown in [Figure 18-81](#).



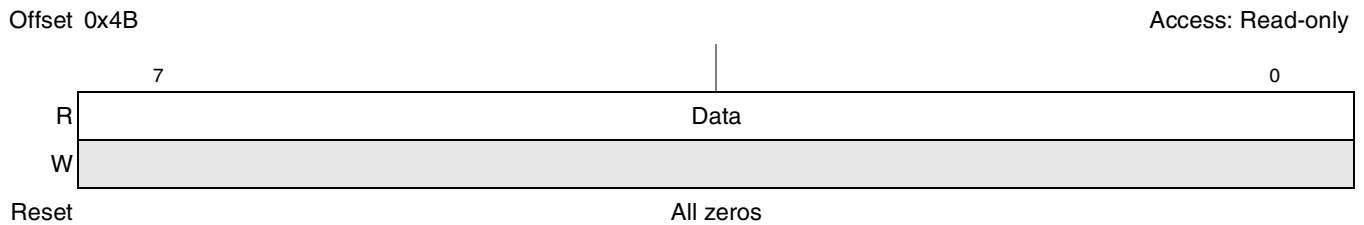
**Figure 18-81. PCI Express Power Management Status and Control Register**

**Table 18-76. PCI Express Status and Control Register Field Description**

Bits	Name	Description
15	PME_STAT	PME Status
14–13	Data Scale	Obtained directly from <i>PCI Express™ Base Specification, Revision 1.0a</i>
12–9	Data Select	Obtained directly from <i>PCI Express™ Base Specification, Revision 1.0a</i>
8	PME_EN	PME Enable
7–2	—	Reserved
1–0	Power State	Power state. Indicates the current power state of the function. 0x00 D0 0x01 D1 0x02 D2 0x03 D3

### 18.3.9.4 PCI Express Power Management Data Register—0x4B

The PCI Express power management data register is shown in [Figure 18-82](#).



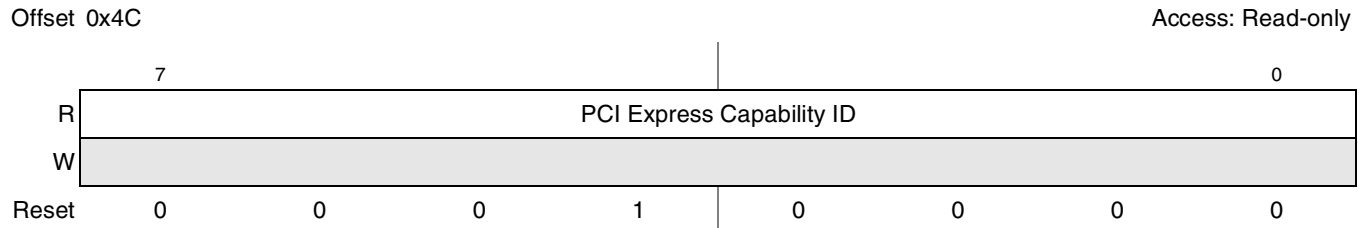
**Figure 18-82. PCI Express Power Management Data Register**

**Table 18-77. PCI Express Power Management Data Register Field Description**

Bits	Name	Description
7–0	Data	Obtained from <i>PCI Express™ Base Specification, Revision 1.0a</i>

### 18.3.9.5 PCI Express Capability ID Register—0x4C

The PCI Express capability ID register is shown in [Figure 18-83](#).



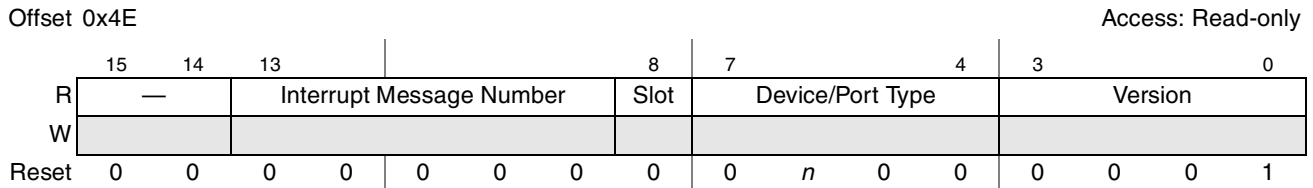
**Figure 18-83. PCI Express Capability ID Register**

**Table 18-78. PCI Express Capability ID Register Field Description**

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

### 18.3.9.6 PCI Express Capabilities Register—0x4E

The PCI Express capabilities register is shown in [Figure 18-84](#).



**Figure 18-84. PCI Express Capabilities Register**

**Table 18-79. PCI Express Capabilities Register Field Description**

Bits	Name	Description
15–14	—	Reserved
13–9	Interrupt Message Number	If this function is allocated more than one MSI interrupt number, then this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register, of this capability structure, are set.
8	Slot	Slot Implemented (RC mode only)
7–4	Device/Port Type	0100 (RC mode) 0000 (EP mode)
3–0	Capability Version	Indicates the defined PCI Express capability structure version number. Must be 1h for this specification.

### 18.3.9.7 PCI Express Device Capabilities Register—0x50

The PCI Express device capabilities register is shown in [Figure 18-85](#).

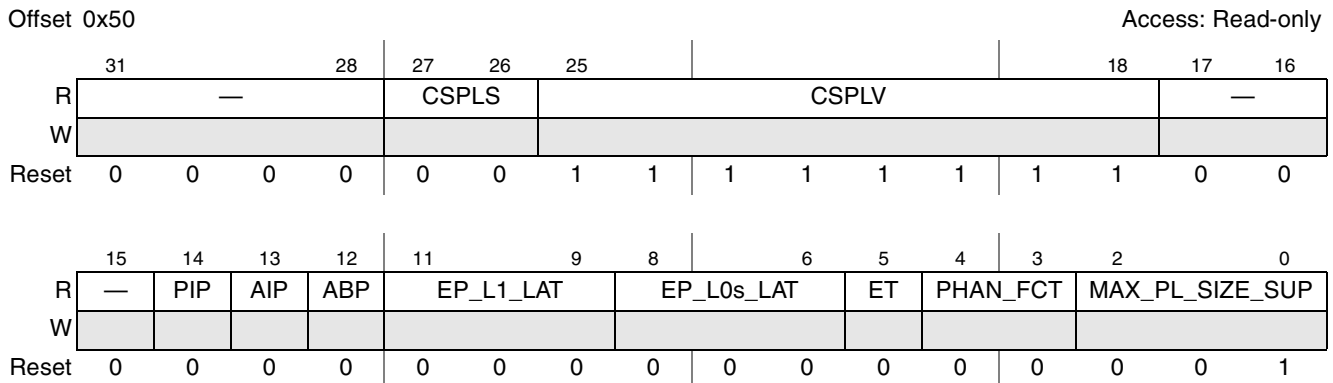


Figure 18-85. PCI Express Device Capabilities Register

Table 18-80. PCI Express Device Capabilities Register Field Description

Bits	Name	Description
31–28	—	Reserved
27–26	CSPLS	Captured Slot Power Limit Scale
25–18	CSPLV	Captured Slot Power Limit Value
17–15	—	Reserved
14	PIP	Power Indicator Present
13	AIP	Attention Indicator Present
12	ABP	Attention Button Present
11–9	EP_L1_LAT	Endpoint L1 Acceptable Latency
8–6	EP_L0s_LAT	Endpoint L0s Acceptable Latency
5	ET	Extended Tag Field Supported
4–3	PHAN_FCT	Phantom Functions Supported
2–0	MAX_PL_SIZE_SUP	Maximum payload size supported. 001 = 256-bytes

### 18.3.9.8 PCI Express Device Control Register—0x54

The PCI Express device control register is shown in [Figure 18-86](#).

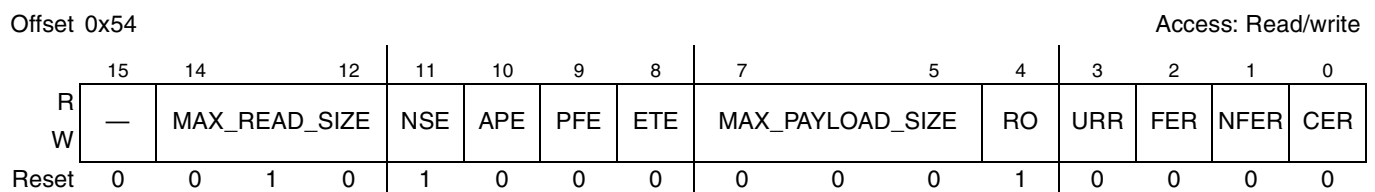


Figure 18-86. PCI Express Device Control Register

**Table 18-81. PCI Express Device Control Register Field Description**

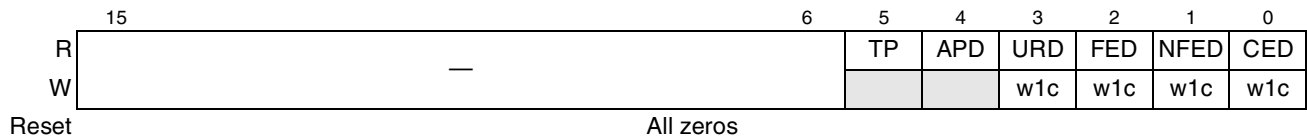
Bits	Name	Description
15	—	Reserved
14–12	MAX_READ_SIZE	Maximum read request size
11	NSE	No snoop enable
10	APE	AUX power PM enable
9	PFE	Phantom functions enable
8	ETE	Extended tag field enable
7–5	MAX_PAYLOAD_SIZE	Maximum payload size
4	RO	Relaxed ordering
3	URR	Unsupported request reporting
2	FER	Fatal error reporting
1	NFER	Non-fatal error reporting
0	CER	Correctable error reporting

### 18.3.9.9 PCI Express Device Status Register—0x56

The PCI Express device status register is shown in [Figure 18-87](#).

Offset 0x56

Access: Mixed



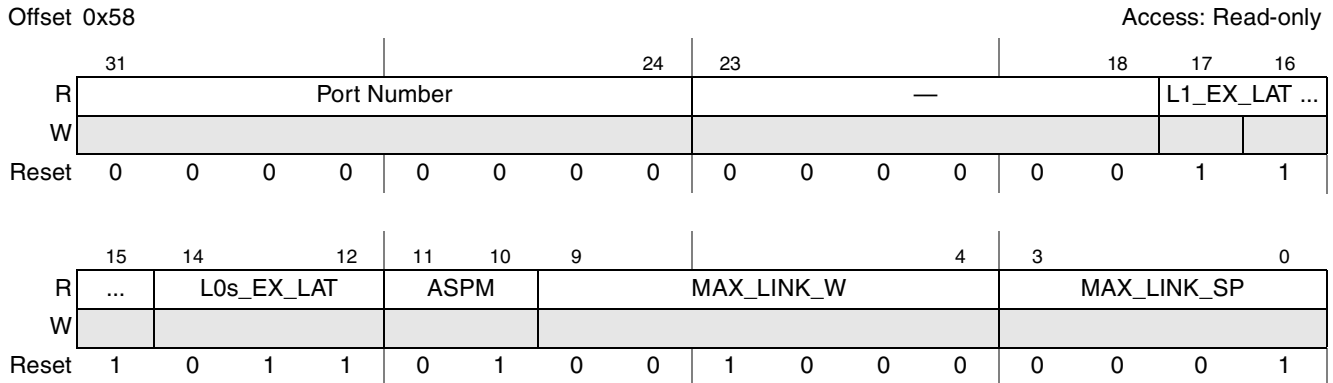
**Figure 18-87. PCI Express Device Status Register**

**Table 18-82. PCI Express Device Status Register Field Description**

Bits	Name	Description
15–6	—	Reserved
5	TP	Transactions pending
4	APD	AUX power detected
3	URD	Unsupported request detected
2	FED	Fatal error detected
1	NFED	Non-fatal error detected
0	CED	Correctable error detected

### 18.3.9.10 PCI Express Link Capabilities Register—0x58

The PCI Express link capabilities register is shown in [Figure 18-88](#).



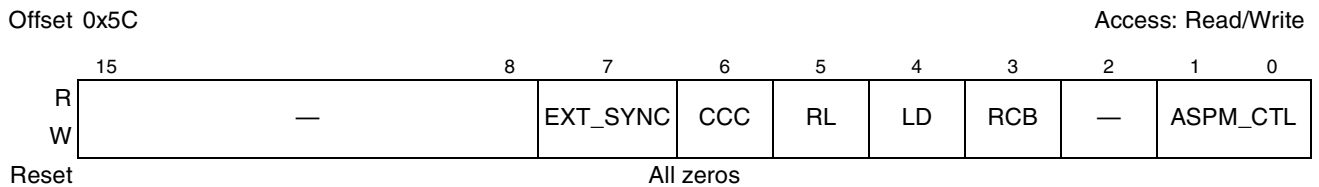
**Figure 18-88. PCI Express Link Capabilities Register**

**Table 18-83. PCI Express Link Capabilities Register Field Description**

Bits	Name	Description
31–24	Port Number	
23–18	—	Reserved
17–15	L1_EX_LAT	L1 exit latency
14–12	L0s_EX_LAT	L0s exit latency
11–10	ASPM	Active state power management (ASPM) Support
9–4	MAX_LINK_W	Maximum link width
3–0	MAX_LINK_SP	Maximum link speed

### 18.3.9.11 PCI Express Link Control Register—0x5C

The PCI Express link control register is shown in [Figure 18-89](#).



**Figure 18-89. PCI Express Link Control Register**

**Table 18-84. PCI Express Link Control Register Field Description**

Bits	Name	Description
15–8	—	Reserved
7	EXT_SYNC	Extended synch
6	CCC	Common clock configuration

**Table 18-84. PCI Express Link Control Register Field Description (continued)**

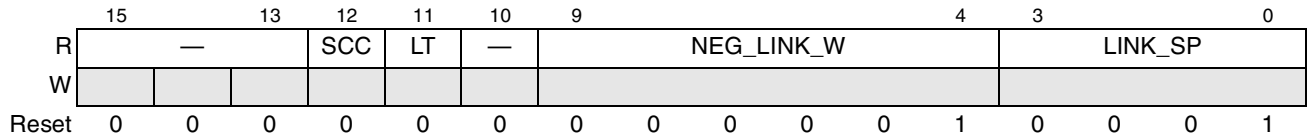
Bits	Name	Description
5	RL	Retrain link
4	LD	Link disable
3	RCB	Read completion boundary
2	—	Reserved
1–0	ASPM_CTL	Active state power management (ASPM) control

### 18.3.9.12 PCI Express Link Status Register—0x5E

The PCI Express link status register is shown in [Figure 18-90](#).

Offset 0x5E

Access: Read-only



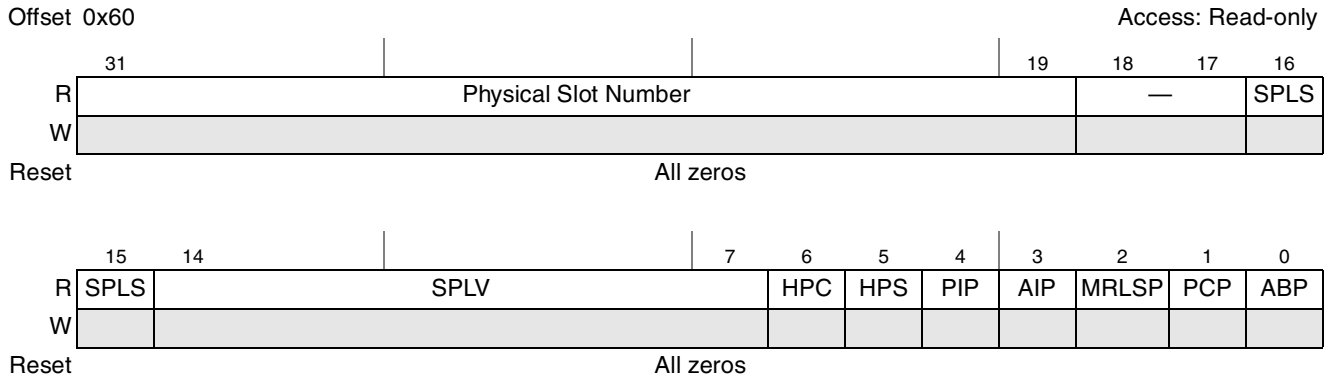
**Figure 18-90. PCI Express Link Status Register**

**Table 18-85. PCI Express Link Status Register Field Description**

Bits	Name	Description
15–13	—	Reserved
12	SCC	Slot clock configuration
11	LT	Link training
10	—	Reserved.
9–4	NEG_LINK_W	Negotiated link width
3–0	LINK_SP	Link speed

### 18.3.9.13 PCI Express Slot Capabilities Register—0x60

The PCI Express slot capabilities register is shown in [Figure 18-91](#).



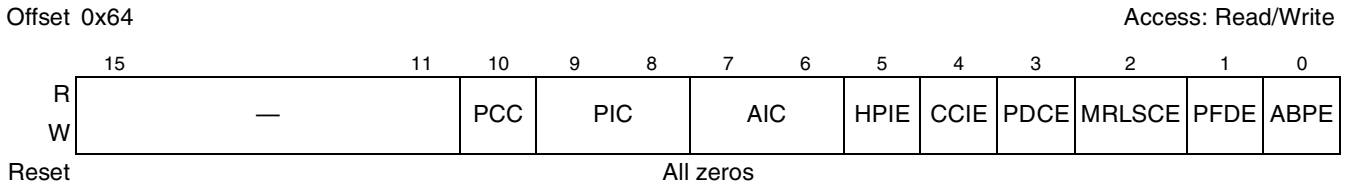
**Figure 18-91. PCI Express Slot Capabilities Register**

**Table 18-86. PCI Express Slot Capabilities Register Field Description**

Bits	Name	Description
31–19	Physical Slot Number	This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18–17	—	Reserved
16–15	SPLS	Slot power limit scale.
14–7	SPLV	Slot power limit value.
6	HPD	Hot plug capable.
5	HPS	Hot plug surprise.
4	PIP	Power indicator present.
3	AIP	Attention indicator present.
2	MRLSP	MRL sensor present.
1	PCP	Power controller present.
0	ABP	Attention button present.

### 18.3.9.14 PCI Express Slot Control Register—0x64

The PCI Express slot control register is shown in [Figure 18-92](#).



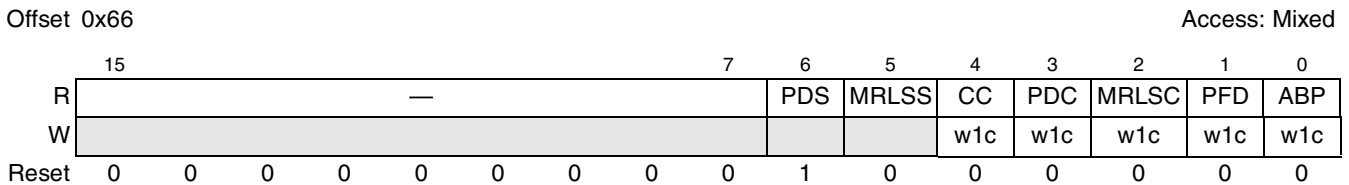
**Figure 18-92. PCI Express Slot Control Register**

**Table 18-87. PCI Express Slot Control Register Field Description**

Bits	Name	Description
15–11	—	Reserved
10	PCC	Power controller control.
9–8	PIC	Power indicator control.
7–6	AIC	Attention indicator control.
5	HPIE	Hot plug interrupt enable.
4	CCIE	Command completed interrupt enable.
3	PDCE	Presence detect changed enable.
2	MRLSCE	MRL sensor changed enable.
1	PFDE	Power fault detected enable.
0	ABPE	Attention button pressed enable.

### 18.3.9.15 PCI Express Slot Status Register—0x66

The PCI Express slot status register is shown in [Figure 18-93](#).



**Figure 18-93. PCI Express Slot Status Register**

**Table 18-88. PCI Express Slot Status Register Field Descriptions**

Bits	Name	Description
15–7	—	Reserved
6	PDS	Presence detect state. This bit indicates the presence of a card in the slot. 0 Slot empty 1 Card is present



**Table 18-88. PCI Express Slot Status Register Field Descriptions (continued)**

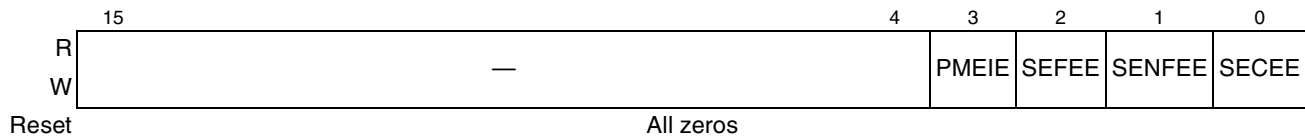
Bits	Name	Description
5	MRLSS	MRL sensor state. 0 MRL closed 1 MRL open
4	CC	Command completed.
3	PDC	Presence detect changed.
2	MRLSC	MRL sensor changed.
1	PFD	Power fault detected.
0	ABP	Attention button pressed.

### 18.3.9.16 PCI Express Root Control Register (RC Mode Only)—0x68

The PCI Express root control register is shown in [Figure 18-94](#).

Offset 0x68

Access: Read/Write



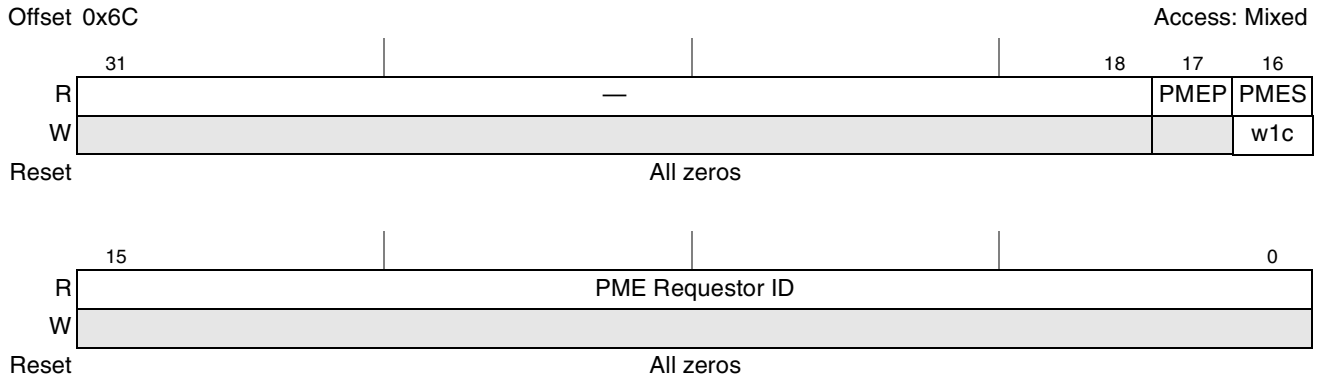
**Figure 18-94. PCI Express Root Control Register**

**Table 18-89. PCI Express Root Control Register Field Description**

Bits	Name	Description
15–4	—	Reserved
3	PMEIE	PME interrupt enable.
2	SEFEE	System error on fatal error enable.
1	SENFEE	System error on non-fatal error enable.
0	SECEE	System error on correctable error enable.

### 18.3.9.17 PCI Express Root Status Register (RC Mode Only)—0x6C

The PCI Express root status register is shown in [Figure 18-95](#).



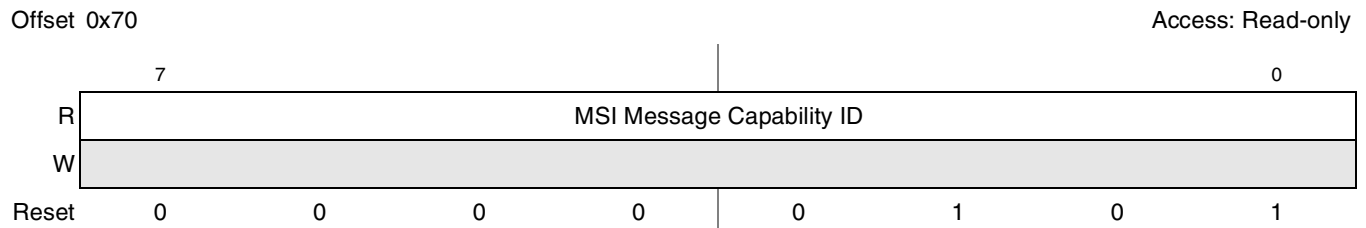
**Figure 18-95. PCI Express Root Status Register**

**Table 18-90. PCI Express Root Status Register Field Description**

Bits	Name	Description
31–18	—	Reserved
17	PMEP	PME pending.
16	PMES	PME status.
15–0	PME Requestor ID	PME requestor ID.

### 18.3.9.18 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70

The PCI Express MSI message capability ID register is shown in [Figure 18-96](#).



**Figure 18-96. PCI Express Capability ID Register**

**Table 18-91. PCI Express Capability ID Register Field Description**

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

### 18.3.9.19 PCI Express MSI Message Control Register (EP Mode Only)—0x72

The PCI Express MSI message control register is shown in [Figure 18-97](#).

Offset 0x72

Access: Mixed

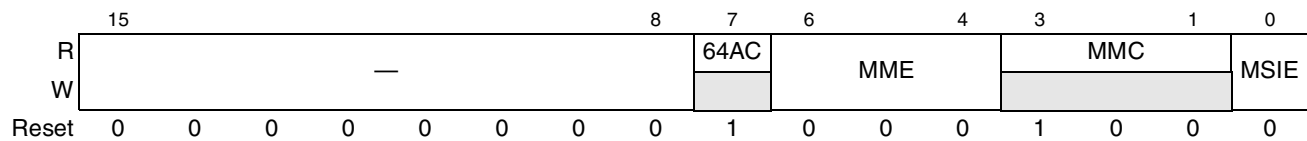


Figure 18-97. PCI Express MSI Message Control Register

Table 18-92. PCI Express MSI Message Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	64AC	64-bit address capable.
6–4	MME	Multiple message enable.
3–1	MMC	Multiple message capable.
0	MSIE	MSI enable.

### 18.3.9.20 PCI Express MSI Message Address Register (EP Mode Only)—0x74

The PCI Express MSI message address register is shown in [Figure 18-98](#).

Offset 0x74

Access: Read/Write

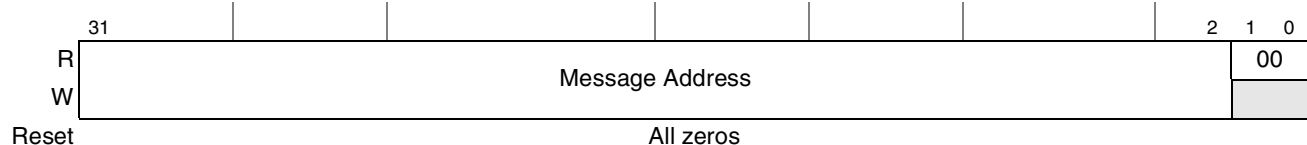


Figure 18-98. PCI Express MSI Message Address Register

Table 18-93. PCI Express MSI Message Address Register Field Description

Bits	Name	Description
31–2	Message Address	System-specified message address
1–0	00	Always returns 00 on reads; write operations have no effect.

### 18.3.9.21 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78

The PCI Express MSI message upper address register is shown in [Figure 18-99](#).

Offset 0x78

Access: Read/Write

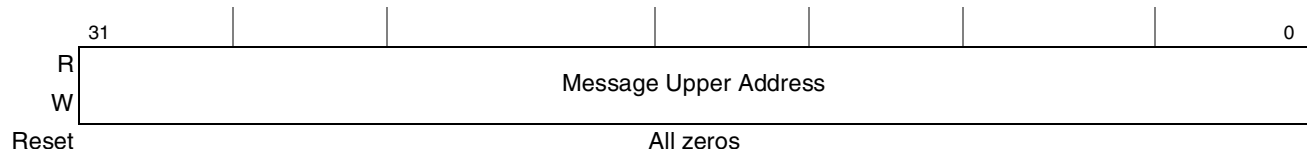


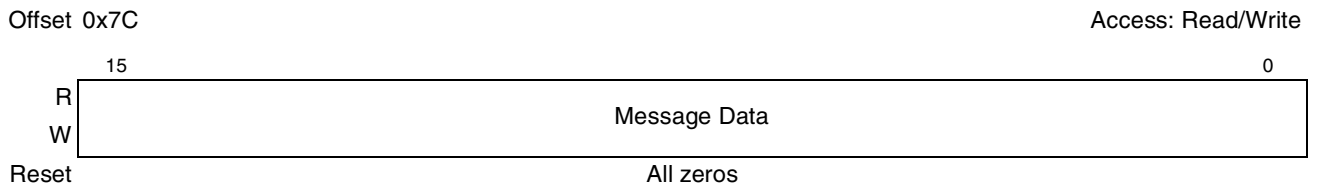
Figure 18-99. PCI Express MSI Message Upper Address Register

**Table 18-94. PCI Express MSI Message Upper Address Register Field Description**

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

### 18.3.9.22 PCI Express MSI Message Data Register (EP Mode Only)—0x7C

The PCI Express MSI message data register is shown in [Figure 18-100](#).



**Figure 18-100. PCI Express MSI Message Data Register**

**Table 18-95. PCI Express MSI Message Data Register Field Description**

Bits	Name	Description
15–0	Message Data	System-specified message.

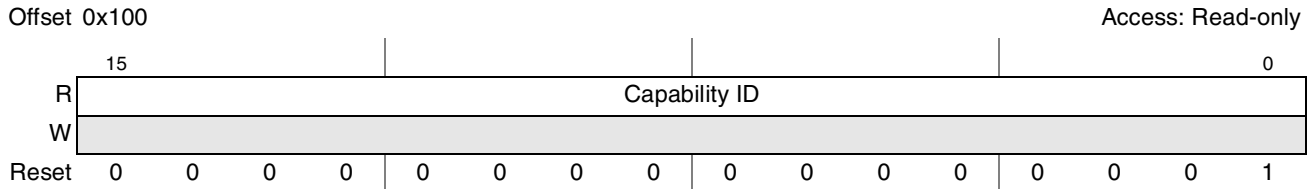
### 18.3.10 PCI Express Extended Configuration Space

Reserved	Address Offset (Hex)		
PCI Compatible Configuration Header (See Section 18.3.8, "PCI Compatible Configuration Headers," for more information.)	000 03F		
PCI-Compatible Device-Specific Configuration Space (See Section 18.3.9, "PCI Compatible Device-Specific Configuration Space," for more information.)	040 0FF		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">Next Capability Offset (NULL)/Capability Version</td> <td style="width: 50%; text-align: center;">Advanced Error Reporting Capability ID</td> </tr> </table>	Next Capability Offset (NULL)/Capability Version	Advanced Error Reporting Capability ID	100
Next Capability Offset (NULL)/Capability Version	Advanced Error Reporting Capability ID		
Uncorrectable Error Status	104		
Uncorrectable Error Mask	108		
Uncorrectable Error Severity	10C		
Correctable Error Status	110		
Correctable Error Mask	114		
Advanced Error Capabilities and Control	118		
Header Log	11C 120 124 128		
Root Error Command	12C		
Root Error Status	130		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">Error Source ID</td> <td style="width: 50%; text-align: center;">Correctable Error Source ID</td> </tr> </table>	Error Source ID	Correctable Error Source ID	134
Error Source ID	Correctable Error Source ID		
	138 3FF		
PCI Express Controller Internal CSRs	400 5A3 5A4		
	FFF		

Figure 18-101. PCI Express Extended Configuration Space

### 18.3.10.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in [Figure 18-102](#).



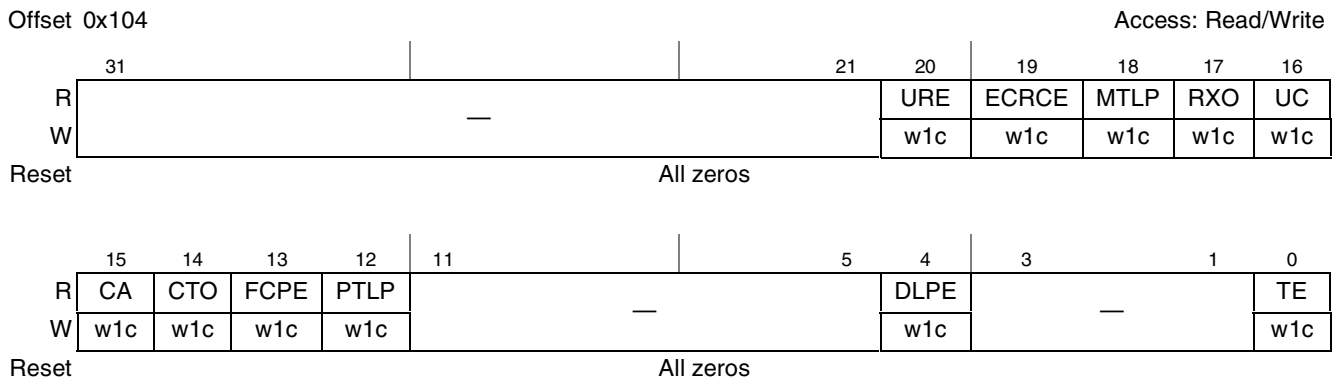
**Figure 18-102. PCI Express Advanced Error Reporting Capability ID Register**

**Table 18-96. PCI Express Advanced Error Reporting Capability ID Register Field Description**

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

### 18.3.10.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in [Figure 18-103](#).



**Figure 18-103. PCI Express Uncorrectable Error Status Register**

**Table 18-97. PCI Express Uncorrectable Error Status Register Field Description**

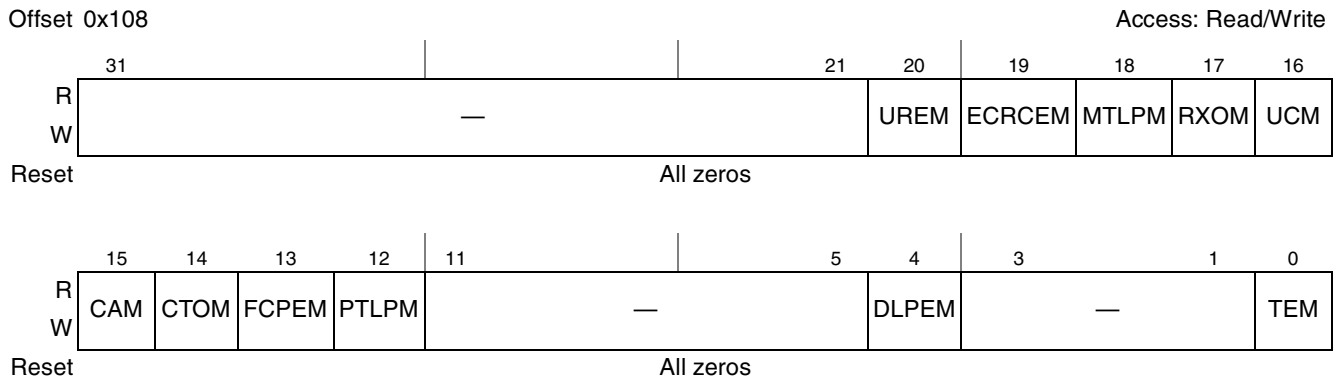
Bits	Name	Description
31–21	—	Reserved
20	URE	Unsupported request error status.
19	ECRCE	ECRC error status.
18	MTLP	Malformed TLP status.
17	RXO	Receiver overflow status.
16	UC	Unexpected completion status.
15	CA	Completer abort status.
14	CTO	Completion timeout status.

**Table 18-97. PCI Express Uncorrectable Error Status Register Field Description (continued)**

Bits	Name	Description
13	FCPE	Flow control protocol error status.
12	PTLP	Poisoned TLP status.
11–5	—	Reserved
4	DLPE	Data link protocol error status.
3–1	—	Reserved
0	TE	Training error status.

### 18.3.10.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in [Figure 18-104](#).



**Figure 18-104. PCI Express Uncorrectable Error Mask Register**

**Table 18-98. PCI Express Uncorrectable Error Mask Register Field Description**

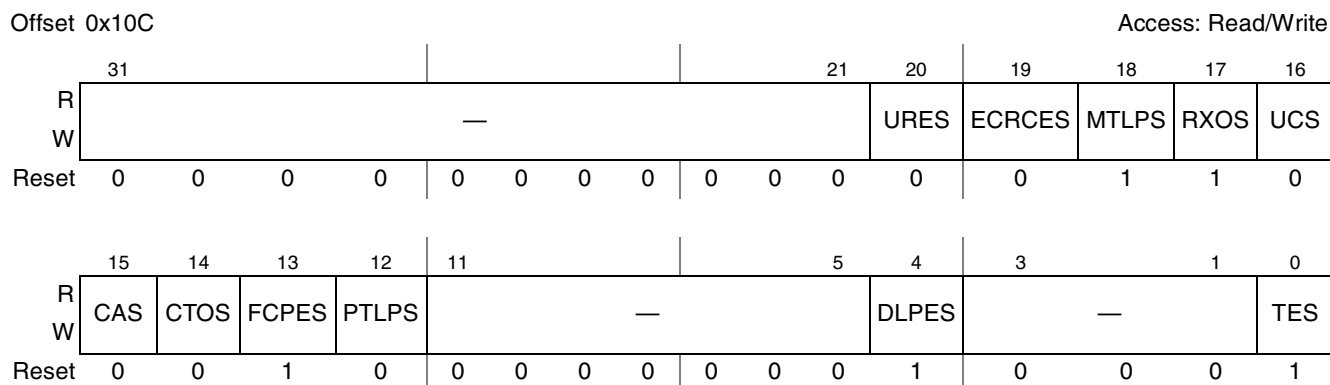
Bits	Name	Description
31–21	—	Reserved
20	UREM	Unsupported request error mask.
19	ECRCEM	ECRC error mask.
18	MTLPM	Malformed TLP mask.
17	RXOM	Receiver overflow mask.
16	UCM	Unexpected completion mask.
15	CAM	Completer abort mask.
14	CTOM	Completion timeout mask.
13	FCPEM	Flow control protocol error mask.
12	PTLPM	Poisoned TLP mask.
11–5	—	Reserved
4	DLPEM	Data link protocol error mask.

**Table 18-98. PCI Express Uncorrectable Error Mask Register Field Description (continued)**

Bits	Name	Description
3–1	—	Reserved
0	TEM	Training error mask.

### 18.3.10.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in [Figure 18-105](#).



**Figure 18-105. PCI Express Uncorrectable Error Severity Register**

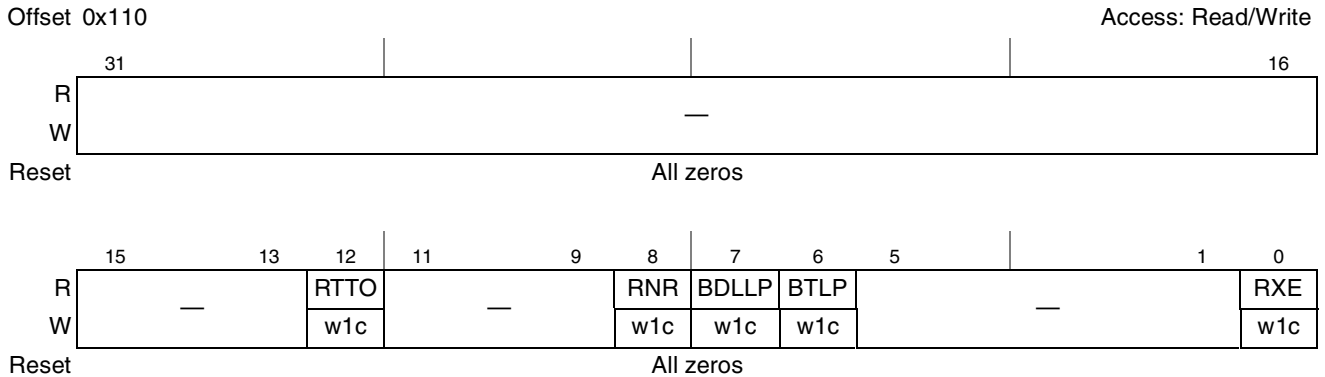
**Table 18-99. PCI Express Uncorrectable Error Severity Register Field Description**

Bits	Name	Description
31–21	—	Reserved
20	URES	Unsupported request error severity.
19	ECRCES	ECRC error severity.
18	MTLPS	Malformed TLP severity.
17	RXOS	Receiver overflow severity.
16	UCS	Unexpected completion severity.
15	CAS	Completer abort severity.
14	CTOS	Completion timeout severity.
13	FCPES	Flow control protocol error severity.
12	PTLPS	Poisoned TLP severity.
11–5	—	Reserved
4	DLPES	Data link protocol error severity.
3–1	—	Reserved
0	TES	Training error severity.



### 18.3.10.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in [Figure 18-106](#).



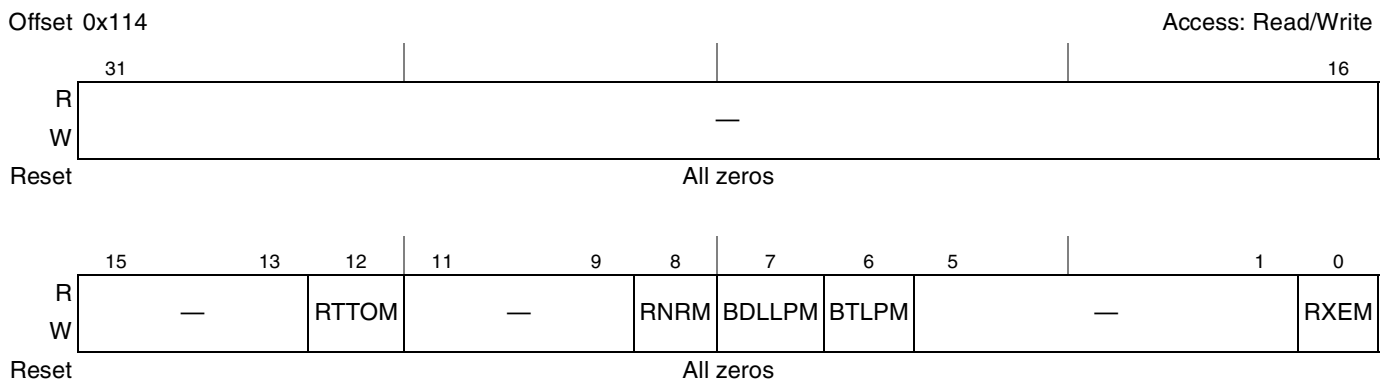
**Figure 18-106. PCI Express Correctable Error Status Register**

**Table 18-100. PCI Express Correctable Error Status Register Field Description**

Bits	Name	Description
31–13	—	Reserved
12	RTTO	Replay timer timeout status
11–9	—	Reserved
8	RNR	REPLAY_NUM Rollover status
7	BDLLP	Bad DLLP status
6	BTLP	Bad TLP status
5–1	—	Reserved
0	RXE	Receiver error status

### 18.3.10.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in [Figure 18-107](#).



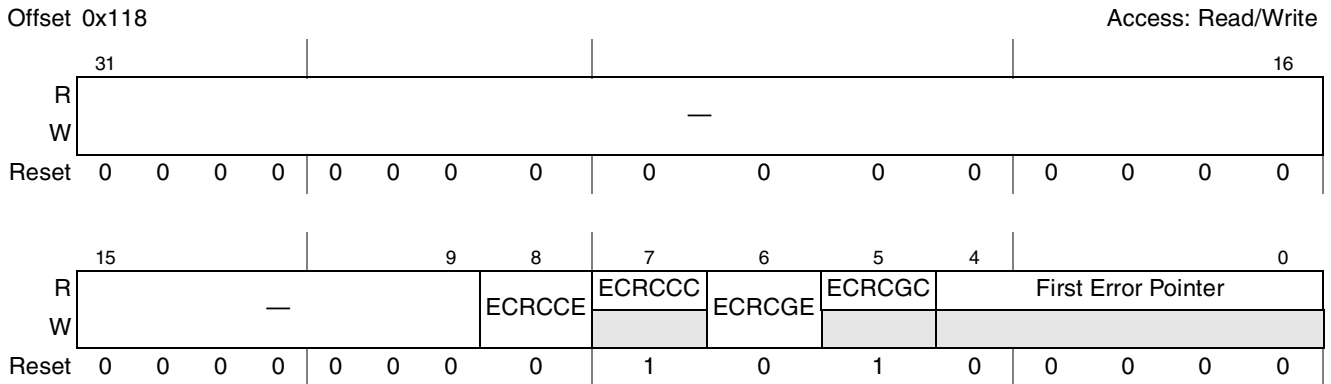
**Figure 18-107. PCI Express Correctable Error Mask Register**

**Table 18-101. PCI Express Correctable Error Mask Register Field Description**

Bits	Name	Description
31–13	—	Reserved
12	RTTOM	Replay timer timeout mask
11–9	—	Reserved
8	RNRM	REPLAY_NUM Rollover mask
7	BDLLPM	Bad DLLP mask
6	BTLPM	Bad TLP mask
5–1	—	Reserved
0	RXEM	Receiver error mask

### 18.3.10.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in [Figure 18-108](#).



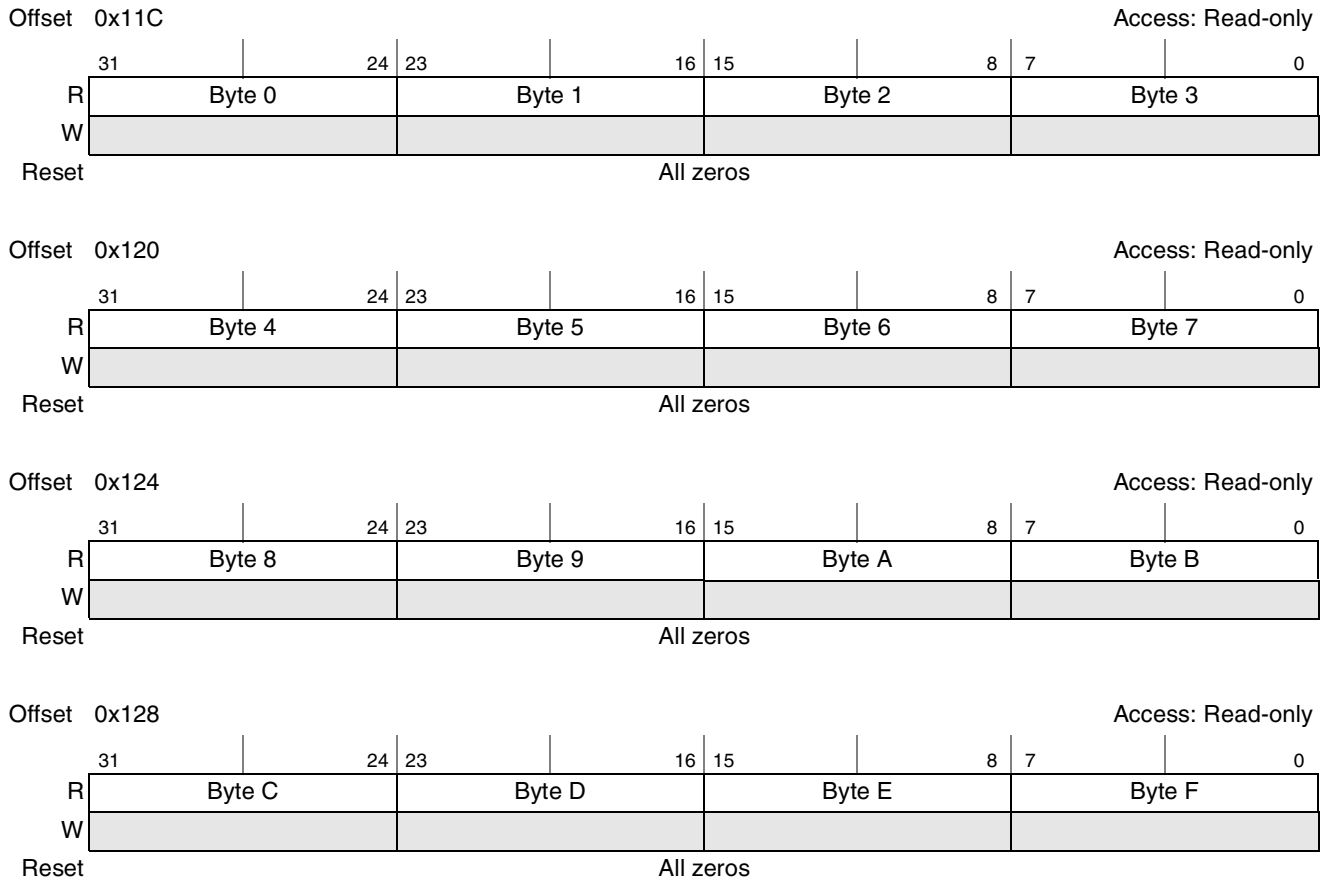
**Figure 18-108. PCI Express Advanced Error Capabilities and Control Register**

**Table 18-102. PCI Express Advanced Error Capabilities and Control Register Field Description**

Bits	Name	Description
31–9	—	Reserved.
8	ECRCCE	ECRC checking enable.
7	ECRCC	ECRC checking capable.
6	ECRCGE	ECRC generation enable.
5	ECRCGC	ECRC generation capable.
4–0	First Error Pointer	The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

### 18.3.10.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in [Figure 18-109](#).



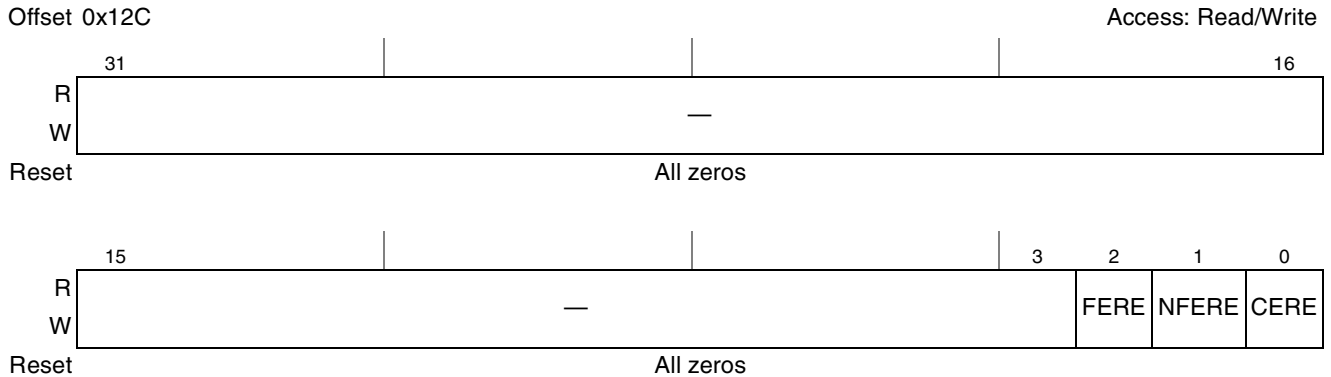
**Figure 18-109. PCI Express Header Log Register**

**Table 18-103. PCI Express Header Log Register Field Description**

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

### 18.3.10.9 PCI Express Root Error Command Register—0x12C

The PCI Express root error command register is shown in [Figure 18-110](#).



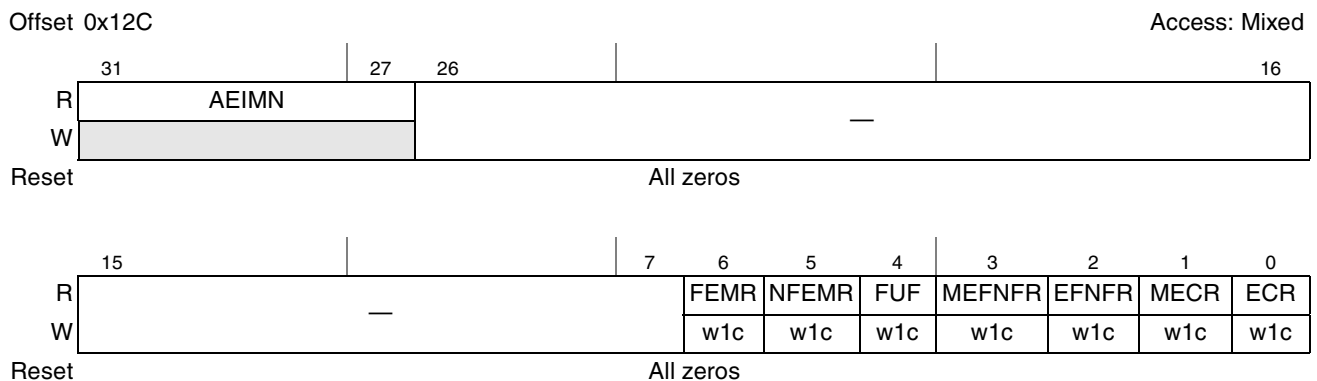
**Figure 18-110. PCI Express Root Error Command Register**

**Table 18-104. PCI Express Root Error Command Register Field Description**

Bits	Name	Description
31–3	—	Reserved
2	FERE	Fatal error reporting enable.
1	NFERE	Non-fatal error reporting enable
0	CERE	Correctable error reporting enable

### 18.3.10.10 PCI Express Root Error Status Register—0x130

The PCI Express root error status register is shown in [Figure 18-111](#).



**Figure 18-111. PCI Express Root Error Command Register**

**Table 18-105. PCI Express Root Error Command Register Field Description**

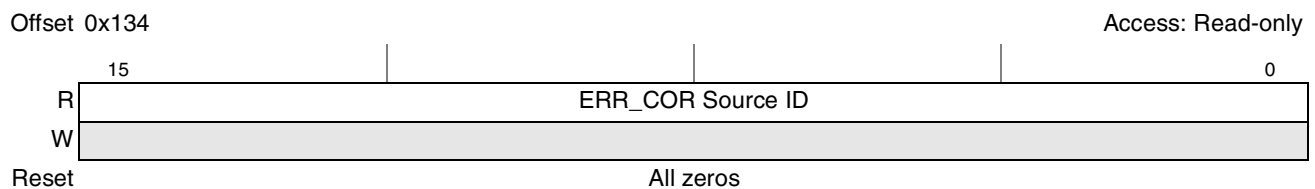
Bits	Name	Description
31–27	AEIMN	Advanced error interrupt message number.
26–7	—	Reserved

**Table 18-105. PCI Express Root Error Command Register Field Description (continued)**

Bits	Name	Description
6	FEMR	Fatal error messages received.
5	NFEMR	Non-fatal error messages received.
4	FUF	First uncorrectable fatal.
3	MEFNFR	Multiple ERR_FATAL/NONFATAL received.
2	EFNFR	ERR_FATAL/NONFATAL received.
1	MECR	Multiple ERR_COR received.
0	ECR	ERR_COR received.

### 18.3.10.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in [Figure 18-112](#).



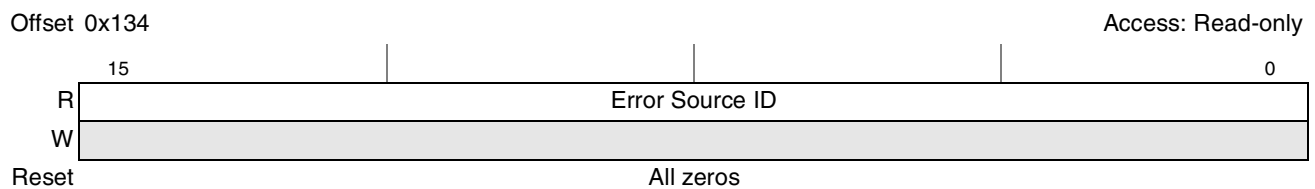
**Figure 18-112. PCI Express Correctable Error Source ID Register**

**Table 18-106. PCI Express Correctable Error Source ID Register Field Description**

Bits	Name	Description
15–0	ERR_COR Source ID	Loaded with the Requestor ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.

### 18.3.10.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in [Figure 18-113](#).



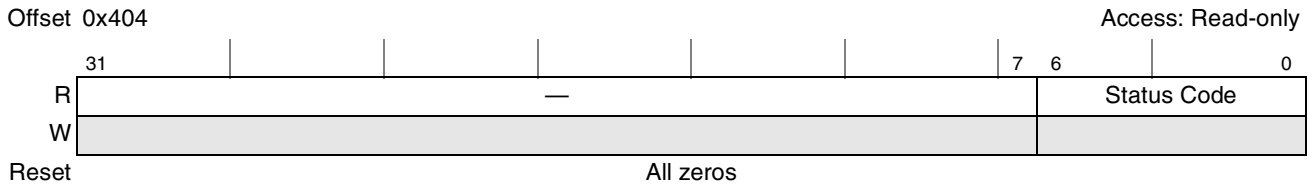
**Figure 18-113. PCI Express Correctable Error Source ID Register**

**Table 18-107. PCI Express Correctable Error Source ID Register Field Description**

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

### 18.3.10.13 LTSSM State Status Register—0x404

The PCI Express link training and status state machine (LTSSM) state status register, shown in [Figure 18-114](#), provides detailed information about link training status. This register is useful for debugging link training failures.



**Figure 18-114. PCI Express LTSSM State Status Register (PEX\_LTSSM\_STAT)**

The fields of the PCI Express LTSSM state status register are described in [Table 18-108](#).

**Table 18-108. PEX\_LTSSM\_STAT Field Descriptions**

Bits	Name	Description
31–7	—	Reserved
6–0	Status code	Status code. See <a href="#">Table 18-109</a> for encodings.

[Table 18-109](#) provides the encodings for the status code field of the PEX\_LTSSM\_STAT register.

**Table 18-109. PEX\_LTSSM\_STAT Status Codes**

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
00	Detect quiet	27	TX L0s FTS; RX L0s FTS
01	Detect active (0)	28	L0 to L1 (0)
02	Detect active (1)	29	L0 to L1 (1)
03	Detect active (2)	2A	L1 entry
04	Polling active (0)	2B	L1 idle (0)
05	Polling active (1)	2C	L1 idle (1)
06	Polling config (0)	2D	L0 to L2 (0)
07	Polling config (1)	2E	L0 to L2 (1)
08	Polling compliance	2F	L2 entry
09	Configuration link width start (0)	30	L2 idle (0)
0A	Configuration link width start (1)	31	L2 idle (1)
0B	Configuration link width accept (0)	32	Recovery lock (0)
0C	Configuration link width accept (1)	33	Recovery lock (1)
0D	Configuration lane number wait (0)	34	Recovery lock (2)

Table 18-109. PEX\_LTSSM\_STAT Status Codes (continued)

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
0E	Configuration lane number wait (1)	35	Recovery cfg (0)
0F	Configuration lane number wait (2)	36	Recovery cfg (1)
10	Configuration lane number wait (3)	37	Recovery idle (0)
11	Configuration lane number accept	38	Recovery idle (1)
12	Configuration complete (0)	39	Recovery to configuration
13	Configuration complete (1)	3A	Recovery cfg to configuration
14	Configuration idle (0)	3F	L0 no training
15	Configuration idle (1)	7F	Detect quiet EI
16	L0	49	Configuration link width start—RC
17	TX L0; RX L0s entry	4A	Configuration link width accept—RC
18	TX L0; RX L0s idle	4B	Configuration lane number wait—RC
19	TX L0; RX L0s fast training sequence (FTS)	4C	Configuration lane number accept—RC
1A	TX L0s entry (0); RX L0	60	Loopback slave active (0)
1B	TX L0s entry (0); RX L0s idle	61	Loopback slave active (1)
1C	TX L0s entry (0); RX L0s FTS	62	Loopback slave exit
1D	TX L0s entry (1); RX L0	68	Hot reset (0)
1E	TX L0s entry (1); RX L0s idle	69	Hot reset (1)
1F	TX L0s entry (1); RX L0s FTS	6A	Hot reset (0)—RC
20	TX L0s idle; RX L0	6B	Hot reset (1)—RC
21	TX L0s idle; RX L0s entry	75	Disabled (0)
22	TX L0s idle; RX L0s idle	71	Disabled (1)
23	TX L0s idle; RX L0s FTS	72	Disabled (2)
24	TX L0s FTS; RX L0	73	Disabled (3)
25	TX L0s FTS; RX L0s entry	74	Disabled (4)
26	TX L0s FTS; RX L0s idle	78	L0 to L1/L2—RC

### 18.3.10.14 PCI Express Controller Core Clock Ratio Register—0x440

The PCI Express controller core clock ratio register, shown in [Figure 18-115](#), is used to program the ratio of the actual PCI Express controller clock frequency to the default controller core frequency (333 MHz). This is required only when a PCI Express controller clock other than the default 333 MHz has to be used.

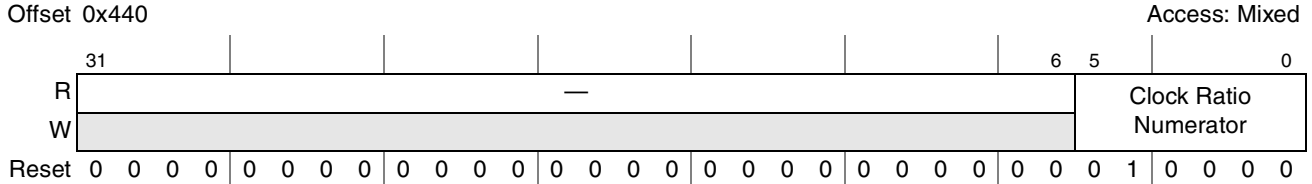


Figure 18-115. PCI Express IP Block Core Clock Ratio Register (PEX\_GCLK\_RATIO)

The fields of the PCI Express IP block core clock ratio register are described in Table 18-110.

Table 18-110. PEX\_GCLK\_RATIO Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5–0	Clock Ratio Numerator	The numerator of the ratio of the actual PCI Express controller clock frequency used to the default core clock frequency of 333 MHz. The denominator of the ratio is fixed at 16. The default value of this register is 0x10 (16 decimal), which corresponds to a ratio of 1:1 (or 16/16)

As an example of programming PEX\_GCLK\_RATIO, consider the case where the actual PCI Express controller clock is 250 MHz, the ratio of the actual clock to the default clock (333 MHz) is 3:4. that is, the default core clock has to be multiplied by the ratio (3/4, which is equivalent to 12/16). So the register has to be programmed with the decimal numerator value 12 or 0x0000\_000C.

**18.3.10.15 PCI Express Power Management Timer Register—0x450**

The PCI Express power management timer register, shown in Figure 18-116, is used to program the time-in values for entering L0s and L1 power management states.

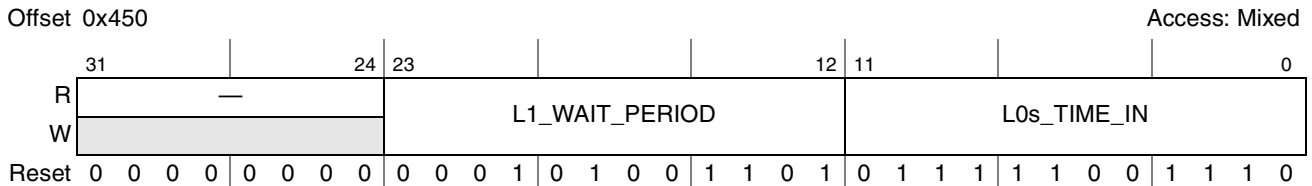


Figure 18-116. PCI Express Power Management Timer Register (PEX\_PM\_TIMER)



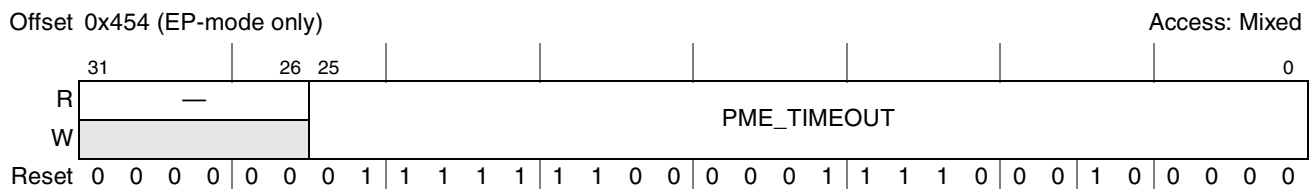
The fields of the PCI Express power management timer register are described in [Table 18-111](#).

**Table 18-111. PEX\_PM\_TIMER Field Descriptions**

Bits	Name	Description
31–24	—	Reserved
23–12	L1_WAIT_PERIOD	Wait period (in PCI Express controller core clock cycles) before entering L1 power state after all functions are in a non-D0 power state. The value is calculated as: Time (in $\mu\text{sec}$ ) $\times$ PCI Express controller core clock frequency (in MHz) The time value must be less than 2 $\mu\text{sec}$ ; the default value (0x14D) is 1 $\mu\text{sec}$ for the default clock frequency of 333 MHz.
11–0	L0s_TIME_IN	Time in value (in PCI Express controller core clock cycles) for entering L0s power state. The value is calculated as: Time (in $\mu\text{sec}$ ) $\times$ PCI Express controller core clock frequency (in MHz) The maximum time value is 7 $\mu\text{sec}$ ; the default value (0x7CE) is 6 $\mu\text{sec}$ for the default clock frequency of 333 MHz.

### 18.3.10.16 PCI Express PME Time-Out Register (EP-Mode Only)—0x454

The PCI Express PME time-out register, shown in [Figure 18-117](#), is used to program the time-out value that the controller uses before re-sending a PME message to the host. If PME is requested by a function and the host does not clear the associated PME\_STAT bit even after this time-out has expired, the PME message is sent again to the host by the PCI Express controller. This register is supported only for EP mode.



**Figure 18-117. PCI Express PME Time-Out Register (PEX\_PME\_TIMEOUT)**

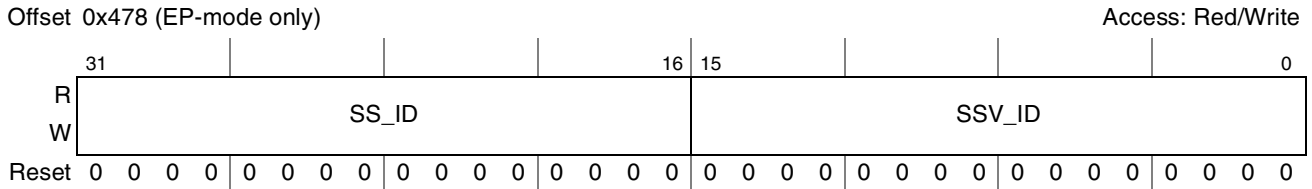
The fields of the PCI Express PME time-out register are described in [Table 18-111](#).

**Table 18-112. PEX\_PME\_TIMEOUT Field Descriptions**

Bits	Name	Description
31–26	—	Reserved
25–0	PME_TIMEOUT	The PME time-out value specifies the interval before PME messages are resent by the controller, provided the PME_STAT bit in the PCI Express power management status and control register (offset 0x48) is not cleared by the host. The value for PME_TIMEOUT is specified in terms of PCI Express controller core clock cycles. The value is calculated as: Time (in $\mu\text{sec}$ ) $\times$ PCI Express controller core clock frequency (in MHz) The minimum time value is 100 msec; the default value (0x1FC1E20) is 100 msec for the default clock frequency of 333 MHz.

### 18.3.10.17 PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478

The PCI Express subsystem vendor ID update register, shown in [Figure 18-118](#), is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.



**Figure 18-118. PCI Express Subsystem Vendor ID Update Register (PEX\_SSVID\_UPDATE)**

The fields of the PCI Express subsystem vendor ID update register are described in [Table 18-113](#).

**Table 18-113. PEX\_SSVID\_UPDATE Field Descriptions**

Bits	Name	Description
31–16	SS_ID	Subsystem ID [15–0] value
15–0	SSV_ID	Subsystem vendor ID [15–0] value

When used as an endpoint, the controller’s initialization software programs the desired subsystem ID and subsystem vendor ID values in PEX\_SSVID\_UPDATE before setting the CFG\_READY bit in the PEX\_CFG\_READY register (see [Section 18.3.10.18, “Configuration Ready Register—0x4B0”](#)). That way, when the host begins system enumeration, the correct values will be present in the Type 0 configuration header.

### 18.3.10.18 Configuration Ready Register—0x4B0

The PCI Express configuration ready register, shown in [Figure 18-119](#), is used to indicate configuration complete status to the transaction layer. The transaction layer handles configuration requests from external hosts only after the CFG\_READY bit is set. All the configuration requests received from external hosts before the CFG\_READY bit is set are completed with configuration request retry status (CRS). The CFG\_READY bit in this register should be set after all relevant configuration registers have been programmed. This makes sure the external host reads the correct capability advertisements during enumeration.

Note that the state of PEX\_CFG\_READY[CFG\_READY] is dependent upon the POR configuration settings described in [Section 4.4.3.5, “Host/Agent Configuration,”](#) and [Section 4.4.3.7, “CPU Boot Configuration.”](#)



**Figure 18-119. PCI Express Configuration Ready Register (PEX\_CFG\_READY)**

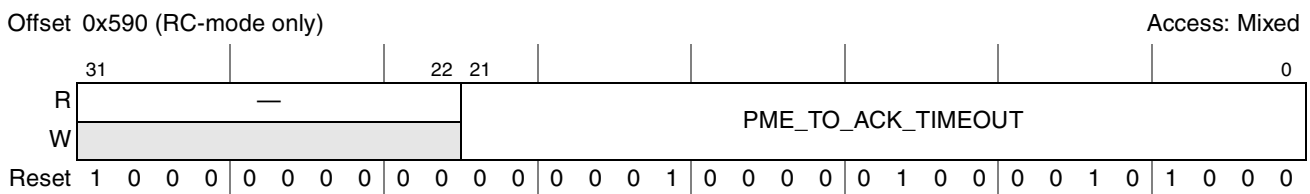
The fields of the PCI Express configuration ready register are described in [Table 18-114](#).

**Table 18-114. PEX\_CFG\_READY Field Descriptions**

Bits	Name	Description
31–1	—	Reserved
0	CFG_READY	Configuration ready 1 The transaction layer will accept inbound configuration requests. 0 The transaction layer will respond to all inbound configuration requests with retry (CRS) Note that the reset state of this bit is determined during POR.

### 18.3.10.19 PME\_To\_Ack Timeout Register (RC-Mode Only)—0x590

The PCI Express PME\_To\_Ack timeout register, shown in [Figure 18-120](#), is used to program the timeout value for a PME\_To\_Ack message response in terms of PCI Express controller core clock cycles.



**Figure 18-120. PCI Express PME\_To\_Ack Timeout Register (PEX\_PME\_TO\_ACK\_TOR)**

The fields of the PCI Express PME\_To\_Ack timeout register are described in [Table 18-115](#).

**Table 18-115. PEX\_PME\_TO\_ACK\_TOR Field Descriptions**

Bits	Name	Description
31–22	—	Reserved
21–0	PME_TO_ACK_TIMEOUT	After a PME_Turn_Off message is broadcast by the RC, the power management module waits for the duration of the PME_To_Ack timeout interval to receive a PME_To_Ack message from the downstream device. If the Ack message is not received within this interval, the power manager indicates that it is safe to switch off power, since timeout has occurred. The value is calculated as: Time (in $\mu$ sec) $\times$ PCI Express controller core clock frequency (in MHz) The recommended timeout duration is 1 msec to 10 msec to make sure that the downstream devices get enough time to prepare for power-off condition.

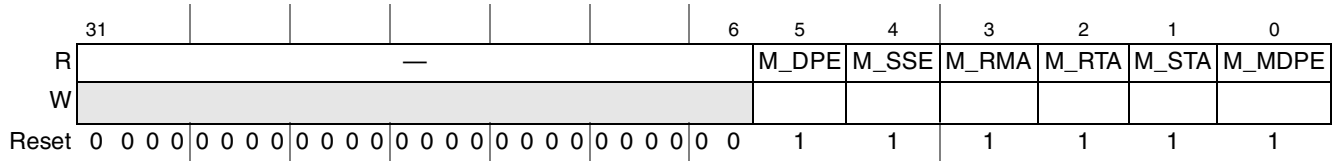
### 18.3.10.20 Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0

The PCI Express secondary status interrupt mask register, shown in [Figure 18-121](#), can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See [Section 18.3.8.3.8, “PCI Express Secondary Status Register—0x1E,”](#) for more information. By default, interrupt generation due to secondary status errors is disabled.

PCI Express Interface Controller

Offset 0x5A0 (RC-mode only)

Access: Mixed



**Figure 18-121. PCI Express PCI Interrupt Mask Register (PEX\_SS\_INTR\_MASK)**

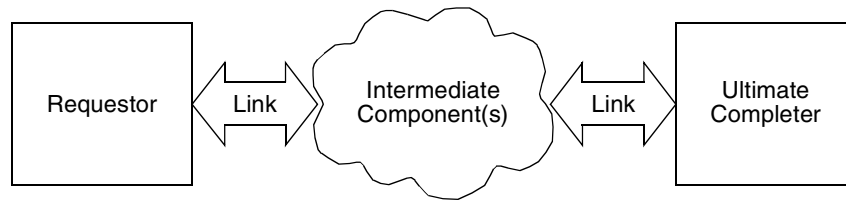
The fields of the PCI Express secondary status interrupt mask register are described in [Table 18-116](#).

**Table 18-116. PEX\_SS\_INTR\_MASK Field Descriptions**

Bits	Name	Description
31–6	—	Reserved
5	M_DPE	Mask detected parity error
4	M_SSE	Mask signaled system error
3	M_RMA	Mask received master abort
2	M_RTA	Mask received target abort
1	M_STA	Mask signaled target abort
0	M_MDPE	Mask master data parity error

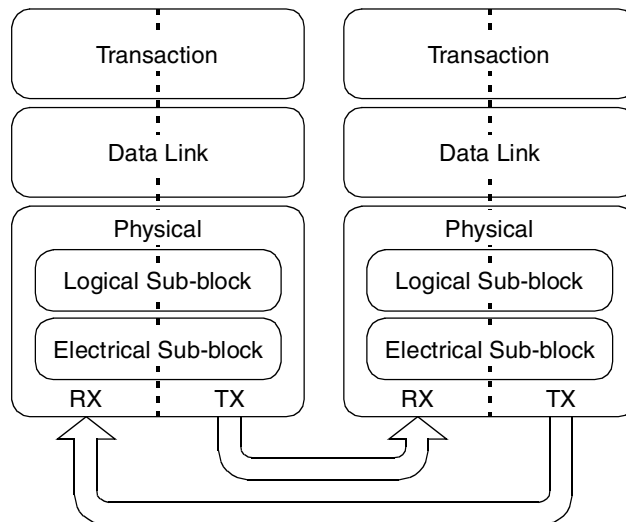
## 18.4 Functional Description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds. Usually the requests and responses occur through a network of links, but to the requestor and to the completer, the intermediate components are transparent.



**Figure 18-122. Requestor/Completer Relationship**

Each PCI device is divided into two halves—transmit (TX) and receive (RX), and each of these halves is further divided into three layers—transaction, data link, and physical—as shown in [Figure 18-123](#).



**Figure 18-123. PCI Express High-Level Layering**

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs), and each subsequent layer adds the necessary encodings and framing—as shown in [Figure 18-124](#). As packets are received, they are decoded and processed by the same layers but in reverse order, so they may be processed by the layer or by the device application software.

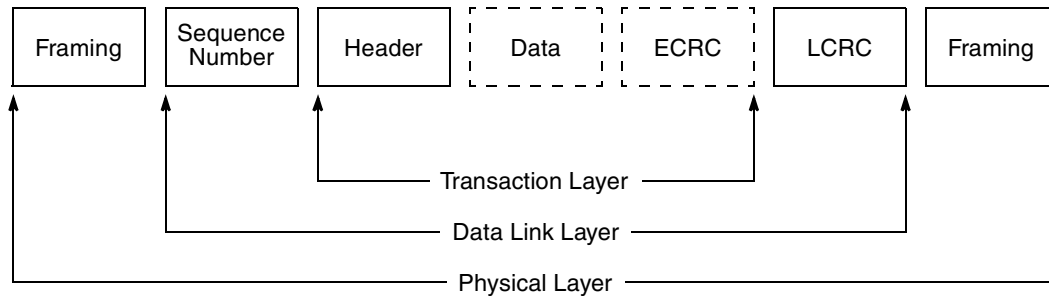


Figure 18-124. PCI Express Packet Flow

## 18.4.1 Architecture

This section describes some of the implementation details of the PCI Express controller.

### 18.4.1.1 PCI Express Transactions

Table 18-117 contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 18-117. PCI Express Transactions

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
Mrd	Yes	Yes	Memory Read Request
MRdLk	No	No	Memory Read Lock. As a target, CplLk with UR status is returned.
MWr	Yes	Yes	Memory Write Request to memory-mapped PCI-Express space
IORd	Yes (RC only)	No	I/O Read request. As a target, Cpl with UR status is returned
IOWr	Yes (RC only)	No	I/O Write Request. As a target, Cpl with UR status is returned
CfgRd0	Yes (RC only)	Yes	Configuration Read Type 0
CfgWr0	Yes (RC only)	Yes	Configuration Write Type 0
CfgRd1	Yes (RC only)	No	Configuration Read Type 1. As a target, Cpl with UR status is returned.
CfgWr1	Yes (RC only)	No	Configuration Write Type 1. As a target, Cpl with UR status is returned.
Msg	Yes	Yes	Message Request
MsgD	Yes (RC only)	Yes (EP only)	Message Request with Data payload. Note that Set_Slot_Power_Limit is the only message with data that is supported and then only when the controller is an initiator and in RC mode or a target and in EP mode.
Cpl	Yes	Yes	Completion without Data
CplD	Yes	Yes	Completion with Data

Table 18-117. PCI Express Transactions (continued)

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
CplLk	No	Yes	Completion for Locked Memory Read without Data. The only time that CplLk is returned with UR status is when the controller receives a MRdLk command.
CpIDLk	No	No	Completion for Locked Memory Read with Data

### 18.4.1.2 Byte Ordering

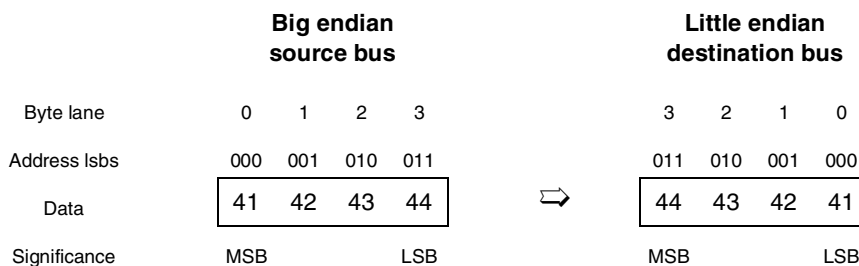
Whenever data must cross a bridge between two busses, the byte ordering of data on the source and destination buses must be considered. The internal platform bus of this device is inherently big endian and the PCI Express bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge—address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its byte ordering policy.

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register. This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

Figure 18-125 shows the transfer of a 4-byte scalar, 0x4142\_4344, from a big endian source across an address invariant bridge to a little endian destination.



**Figure 18-125. Address Invariant Byte Ordering—4 bytes Outbound**

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

Figure 18-126 shows data flowing the other way, from a little endian source to a big endian destination.

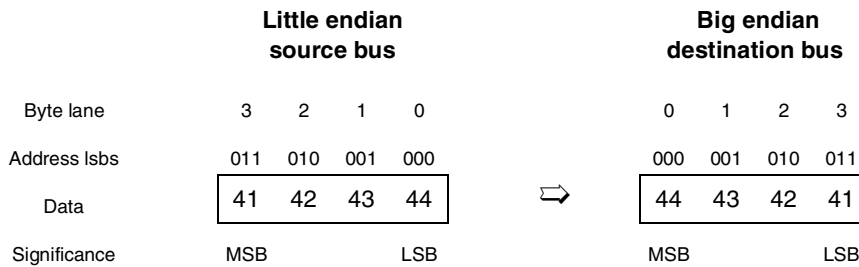


Figure 18-126. Address Invariant Byte Ordering—4 bytes Inbound

Figure 18-127 shows an outbound transfer of an 8-byte scalar, 0x5455\_1617\_CDCE\_2728, using address invariance.

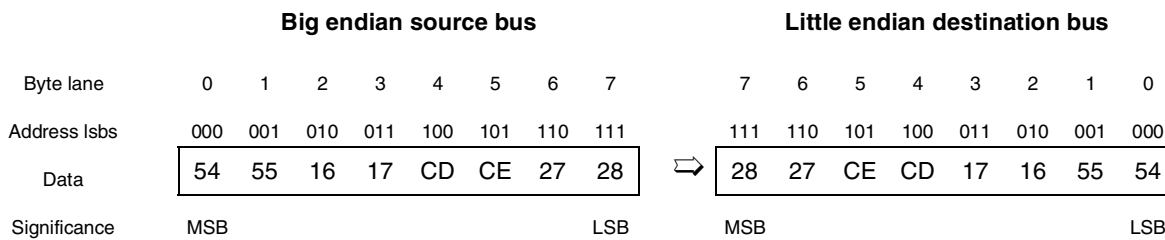


Figure 18-127. Address Invariant Byte Ordering—8 bytes Outbound

Figure 18-128 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

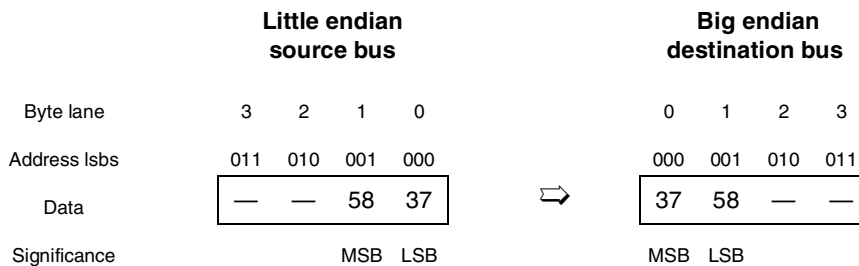


Figure 18-128. Address Invariant Byte Ordering—2 bytes Inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

### 18.4.1.2.1 Byte Order for Configuration Transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI Express specification defines PCI Express configuration registers as little endian. All accesses to the PCI Express configuration port, PEX\_CONFIG\_DATA, including the those targeting the internal PCI Express configuration registers, use the address invariance policy as shown in Figure 18-129. Therefore, software must access PEX\_CONFIG\_DATA with little-endian formatted data—either by using the **lwbrx/stwbrx** instructions or by manipulating the data before writing to and after reading from PEX\_CONFIG\_DATA.



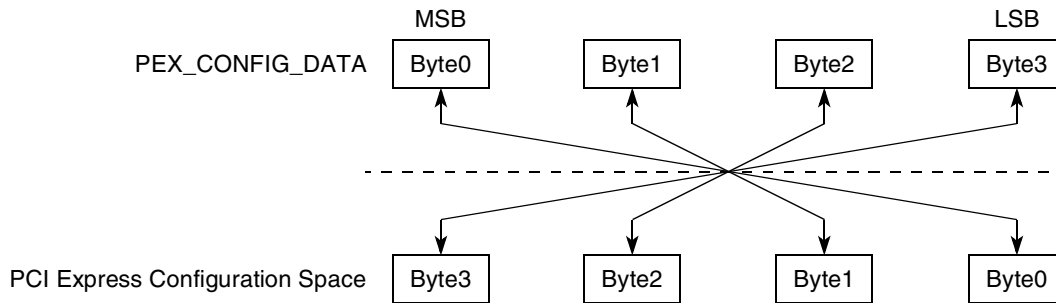


Figure 18-129. PEX\_CONFIG\_DATA Byte Ordering

### 18.4.1.3 Transaction Ordering Rules

In general, transactions are serviced in the order that they are received. However, transactions can be reordered as they are sent due to a stalled condition such as a full internal buffer. The following are the ordering rules for sending the next outstanding request:

- A posted request can and will bypass all other transactions except another posted request.
- A completion can and will only bypass non-posted. It can and will bypass posted requests only if the relaxed ordering (RO) bit is set.
- A non-posted request cannot bypass posted or other non-posted requests, but it can bypass a completion if the relaxed ordering (RO) bit is set.

### 18.4.1.4 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. The Fmt[0] field in the PCI Express TLP header for a 32-bit address packet is 0; a 64-bit address packet has a Fmt[0] = 1. The PCI Express TLP header for a memory read transaction has Type[4:0] = 00000 and Fmt[1] = 0. A memory write transaction has Type[4:0] = 00000 and Fmt[1] = 1. As an initiator, the controller is capable of sending 32- or 64-bit memory packets. Any transaction from the internal platform that (after passing through the translation mechanism) has a translated address greater than 4G is sent as a 64-bit memory packet. Otherwise, a 32-bit memory packet is sent. As a target device, the controller is capable of decoding 32- or 64-bit memory packets. This is done through two 32-bit inbound windows and two 64-bit inbound windows. All inbound addresses are translated to 36-bit internal platform addresses.

### 18.4.1.5 I/O Space Addressing

The controller does not support I/O transactions as a target. As an initiator, the controller can send I/O transactions in RC mode only. This can be done by programming one of the outbound translation window's attribute to send I/O transactions. All I/O transactions only access 32-bit address I/O space. The PCI Express TLP header for an I/O read transaction has Type[4:0] = 00010 and Fmt[1] = 0. The PCI Express TLP header for an I/O write transaction has Type[4:0] = 00010 and Fmt[1] = 1.

### 18.4.1.6 Configuration Space Addressing

As an initiator, the controller supports both type 0 and type 1 configuration cycles when configured in RC mode. There are two methods of generating a configuration transaction; refer to [Section 18.3.7, “PCI Express Configuration Space Access,”](#) for more information. A configuration transaction can hit into the controller’s internal configuration space, it can be sent out on the PCI Express link, or it can be internally terminated. The PCI Express TLP header for a type 0 configuration read transaction has Type[4:0] = 00100 and Fmt[1] = 0; the PCI Express TLP header for a type 0 configuration write transaction has Type[4:0] = 00100 and Fmt[1] = 1. The PCI Express TLP header for a type 1 configuration read transaction has Type[4:0] = 00101 and Fmt[1] = 0; the PCI Express TLP header for a type 1 configuration write transaction has Type[4:0] = 00101 and Fmt[1] = 1. Note that all configuration transactions sent on PCI Express require a response regardless whether they are read or a write configuration transactions.

The controller does not generate configuration transactions in EP mode. Only inbound configuration transactions are supported in EP mode.

### 18.4.1.7 Serialization of Configuration and I/O Writes

Configuration and I/O writes originating from the PCI Express outbound ATMUs are serialized by the controller. The logic after issuing a configuration write or IO write will not issue any new transactions until the outstanding configuration or I/O write is finished. This means that an acknowledgement packet from the link partner in the form of a CpL TLP packet must be seen or the transaction has timed out. If the CpL packet contains a CRS status, then the logic will re-issue the configuration write transaction. It will keep retrying the request until either a status other than CRS is returned or the transaction times out. Note that configuration writes originating from the PCI Express configuration access registers (PEX\_CONFIG\_ADDR/PEX\_CONFIG\_DATA) are not serialized.

Note that it is possible for outbound configuration read request to be requeued and be placed at the end of the request queue due to CRS condition.

### 18.4.1.8 Messages

Software message generation is supported in both RC and EP modes.

#### 18.4.1.8.1 Outbound ATMU Message Generation

Software can choose to send a message by programming  $PEXOWAR_n[WTT] = 0x5$ . A message is sent by writing a 4-byte transaction in big-endian format that hits in an outbound window configured to send messages.

Part of the 4-byte data is used to store information such as message code and routing information. [Table 18-118](#) describes the message data format.

**Table 18-118. Internal Platform (OCeaN) Message Data Format**

Bits	Name	Reset Value	Description
0–15	—	—	Reserved
16–18	Routing	x	Routing mechanism. Contains the message’s routing information
19–23	—	—	Reserved
24–31	Code	x	Message code. Contains the actual message type to be sent.

In addition to the outbound ATMU, the PEX PM Command register also provides the capability to send PME\_Turn\_Off message or PM\_PME message by setting bits 31 or 29. See [Section 18.3.3.4, “PCI Express Power Management Command Register \(PEX\\_PMCR\),”](#) on page 18-17 for more information.

[Table 18-119](#) provides a complete list of supported outbound messages depending on whether RC or EP is configured.

**Table 18-119. PCI Express ATMU Outbound Messages**

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Assert_INTA	0010 0000	100	N/A	Yes	Sent upstream by endpoint
Assert_INTB	0010 0001	100	N/A	Yes	Sent upstream by endpoint
Assert_INTC	0010 0010	100	N/A	Yes	Sent upstream by endpoint
Assert_INTD	0010 0011	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTA	0010 0100	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTB	0010 0101	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTC	0010 0110	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTD	0010 0111	100	N/A	Yes	Sent upstream by endpoint
PM_Active_State_Nak	0001 0100	100	Yes	N/A	Terminate at receiver
PM_PME	0001 1000	000	N/A	Yes	Sent Upstream by PME-requesting component
PME_Turn_Off	0001 1001	011	Yes	N/A	Broadcast Downstream
PM_TO_Ack	0001 1011	101	N/A	Yes	Sent Upstream by Endpoint
ERR_COR	0011 0000	000	N/A	Yes	Sent by component when it detects a correctable error
ERR_NONFATAL	0011 0001	000	N/A	Yes	Sent by component when it detects a Non-fatal, uncorrectable error
ERR_FATAL	0011 0011	000	N/A	Yes	Sent by component when it detects a Fatal, uncorrectable error
Unlock	0000 0000	000	No	N/A	Not supported

**Table 18-119. PCI Express ATMU Outbound Messages (continued)**

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Set_Slot_Power_Limit	0101 0000	100	Yes	N/A	Set Slot Power Limit in Upstream Port
Vendor_Defined Type 0	0111 1110		No	No	Not supported
Vendor_Defined Type 1	0111 1111		No	No	Not supported
Attention_Indicator_On	0100 0001	100	Yes	N/A	Hot-plug message
Attention_Indicator_Blink	0100 0011	100	Yes	N/A	Hot-plug message
Attention_Indicator_Off	0100 0000	100	Yes	N/A	Hot-plug message
Power_Indicator_On	0100 0101	100	Yes	N/A	Hot-plug message
Power_Indicator_Blink	0100 0111	100	Yes	N/A	Hot-plug message
Power_Indicator_Off	0100 0100	100	Yes	N/A	Hot-plug message
Attention_Button_Pressed	0100 1000	100	Yes	N/A	Hot-plug message

### 18.4.1.8.2 Inbound Messages

Table 18-120 provides a complete list of supported inbound messages in RC mode.

**Table 18-120. PCI Express RC Inbound Message Handling**

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	Send to PIC
Assert_INTB	0010 0001	100	Send to PIC
Assert_INTC	0010 0010	100	Send to PIC
Assert_INTD	0010 0011	100	Send to PIC
De-assert_INTA	0010 0100	100	Send to PIC
De-assert_INTB	0010 0101	100	Send to PIC
De-assert_INTC	0010 0110	100	Send to PIC
De-assert_INTD	0010 0111	100	Send to PIC
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	Generate interrupt to PIC if enabled
PME_Turn_Off	0001 1001	011	No action taken
PME_TO_Ack	0001 1011	101	Set PEX_PME_MES_DR[ENL23] bit and generate interrupt to PIC if enabled
ERR_COR	0011 0000	000	Generate interrupt to PIC if enabled
ERR_NONFATAL	0011 0001	000	Generate interrupt to PIC if enabled
ERR_FATAL	0011 0011	000	Generate interrupt to PIC if enabled
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	No action taken

**Table 18-120. PCI Express RC Inbound Message Handling (continued)**

Name	Code[7:0]	Routing[2:0]	Action
Vendor_Defined Type 0	0111 1110		No action taken
Vendor_Defined Type 1	0111 1111		No action taken
Attention_Indicator_On	0100 0001	100	No action taken
Attention_Indicator_Blink	0100 0011	100	No action taken
Attention_Indicator_Off	0100 0000	100	No action taken
Power_Indicator_On	0100 0101	100	No action taken
Power_Indicator_Blink	0100 0111	100	No action taken
Power_Indicator_Off	0100 0100	100	No action taken
Attention_Button_Pressed	0100 1000	100	Set PEX_PME_MES_DR[ABP] bit and send interrupt if enabled.

Table 18-121 provides a complete list of supported inbound messages in EP mode.

**Table 18-121. PCI Express EP Inbound Message Handling**

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	No action taken
Assert_INTB	0010 0001	100	No action taken
Assert_INTC	0010 0010	100	No action taken
Assert_INTD	0010 0011	100	No action taken
Deassert_INTA	0010 0100	100	No action taken
Deassert_INTB	0010 0101	100	No action taken
Deassert_INTC	0010 0110	100	No action taken
Deassert_INTD	0010 0111	100	No action taken
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	No action taken
PME_Turn_Off	0001 1001	011	Set PEX_PME_MES_DR[PTO] bit. Send interrupt if enabled.
PM_TO_Ack	0001 1011	101	No action taken
ERR_COR	0011 0000	000	No action taken
ERR_NONFATAL	0011 0001	000	No action taken
ERR_FATAL	0011 0011	000	No action taken
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.
Vendor_Defined Type 0	0111 1110		No action taken

Table 18-121. PCI Express EP Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Vendor_Defined Type 1	0111 1111		No action taken
Attention_Indicator_On	0100 0001	100	Set PEX_PME_MES_DR[AION] bit. Send interrupt if enabled.
Attention_Indicator_Blink	0100 0011	100	Set PEX_PME_MES_DR[AIB] bit. Send interrupt if enabled.
Attention_Indicator_Off	0100 0000	100	Set PEX_PME_MES_DR[AIOF] bit. Send interrupt if enabled.
Power_Indicator_On	0100 0101	100	Set PEX_PME_MES_DR[PION] bit. Send interrupt if enabled.
Power_Indicator_Blink	0100 0111	100	Set PEX_PME_MES_DR[PIB] bit. Send interrupt if enabled.
Power_Indicator_Off	0100 0100	100	Set PEX_PME_MES_DR[PIOF] bit. Send interrupt if enabled.
Attention_Button_Pressed	0100 1000	100	No action taken

## 18.4.2 Interrupts

Both INTx and message signaled interrupts (MSI) are supported; however there are subtle differences depending on whether the PCI Express controller is configured as an RC or EP device.

### 18.4.2.1 EP Interrupt Generation

#### 18.4.2.1.1 Hardware INTx Message Generation

Hardware INTx message generation is not supported in EP mode.

#### 18.4.2.1.2 Hardware MSI Generation

In EP mode, the PCI Express controller can be configured to automatically generate MSI transactions to the root complex when an interrupt event occurs. The PCI Express controller uses *irq\_out* (an internal version of the IRQ\_OUT signal) to trigger the generation of the MSI. To trigger the MSI, interrupt events must be routed to the *irq\_out* by setting the EP (external pin) bit in the associated Interrupt Destination register in the PIC. Note that the IRQ\_OUT signal should not be used for any other function if it is being used to trigger MSI transactions.

The remote root complex is expected set up the MSI capability structure of all endpoints at system initialization by filling the Message Address and Message Data registers with appropriate values and setting the MSIE bit in the MSI Message Control register.

With the PCI Express controller properly configured, when it detects the leading edge of *irq\_out* going active, it generates a PCI Express memory write transaction to the address specified in the MSI Message Address register (and MSI Message Upper Address register) with a data payload as specified in the MSI Message Data register (with leading zeros appended).

### 18.4.2.1.3 Software INTx Message Generation

Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in [Section 18.4.1.8.1, “Outbound ATMU Message Generation.”](#)

### 18.4.2.1.4 Software MSI Generation

Host software has to set up the MSI capability registers to enable MSI mode, and have the correct values for the MSI address and data register. Then local software has to read the MSI address in the MSI capability register and configure the outbound ATMU window to map the translated address to the MSI address. Software has to determine the number of allocated messages in the MSI capability register and allocates the appropriate data values to use. A write to the ATMU window containing the MSI address with the appropriate data value will generate the desired MSI transaction to the remote RC.

## 18.4.2.2 RC Handling of INTx Message and MSI Interrupt

### 18.4.2.2.1 INTx Message Handling

MSIs are the preferred interrupt signaling mechanism for PCI Express. However, in RC mode, the PCI Express controller supports the INTx virtual-wire interrupt signaling mechanism (as described in the PCI Express specification). Whenever the controller receives an inbound INTx (INTA, INTB, INTC, or INTD) asserted or negated message, it asserts or negates an equivalent internal INTx signal (*inta*, *intb*, *intc*, or *intd*) to the PIC.

The internal INTx signals from the PCI Express controller are logically combined with the interrupt request (IRQ<sub>n</sub>) input signals so that they share the same interrupt controlled by the associated EIVPR<sub>n</sub> and EIDR<sub>n</sub> registers in the PIC. Refer to [Chapter 10, “Programmable Interrupt Controller,”](#) for more information about handling of PCI Express INTx interrupts and the external interrupt request (IRQ<sub>n</sub>) signals.

If a PCI Express INTx interrupt is being used, then the PIC must be configured so that external interrupts are active-low (EIVPR<sub>n</sub>[P] = 0), and level-sensitive (EIVPR<sub>n</sub>[S] = 1).

### 18.4.2.2.2 MSI Handling

An inbound MSI cycle must hit into the PCSRBAR window with the address offset that points to the MSIIR register in the PIC. Note that it is the responsibility of the host software to configure each EP’s MSI capability registers such that an MSI cycle generated from the EP device is routed to the MSIIR register in the PIC and for the appropriate interrupt to be generated to the core.

## 18.4.3 Initial Credit Advertisement

To prevent overflowing of the receiver’s buffers and for ordering compliance purposes, the transmitter cannot send transactions unless it has enough flow control (FC) credits to send. Each device maintains an FC credit pool. The FC information is conveyed between the two link partners by DLLPs during link training (initial credit advertisement). The transaction layer performs the FC accounting functions. One FC unit is four DWs (16-bytes) of data.

**Table 18-122. Initial Credit Advertisement**

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	4
PD (Memory Write, Message Write)	$(256/16) \times 4 = 64$
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	8
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

## 18.4.4 Power Management

All device power states are supported with the exception of D3cold with Vaux. In addition, all link power states are supported with the exception of L2 states. Only L0s ASPM mode is supported if enabled by configuring the Link Control register's bits 1–0 in configuration space. Note that there is no power saving in the controller when the device is put into a non-D0 state. The only power saving is the I/O drivers when the controller is put into a non-L0 link state.

**Table 18-123. Power Management State Supported**

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffics are stalled. All inbound traffics will be thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message via the PEX Power Management Command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffics will be thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message via the PEX Power Management Command register.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffics are stalled. All inbound traffics will be thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message via the PEX Power Management Command register. Note that if a transition of D3hot->D0 occurs, a reset is performed to the controller's configuration space. In addition, link training will restart.
D3cold	L3	Completely off.

### 18.4.4.1 L2/L3 Ready Link State

The L2/L3 Ready link state is entered after the EP device is put into a D3hot state followed by a PME\_Turn\_Off/PME\_TO\_Ack message handshake protocol. Exiting this state requires a POR reset or a WAKE signal from the EP device. The PCI Express controller (in EP mode) does not support the



generation of beacon; therefore, as an alternative, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a  $\overline{\text{WAKE}}$  signal, as shown in Figure 18-130.

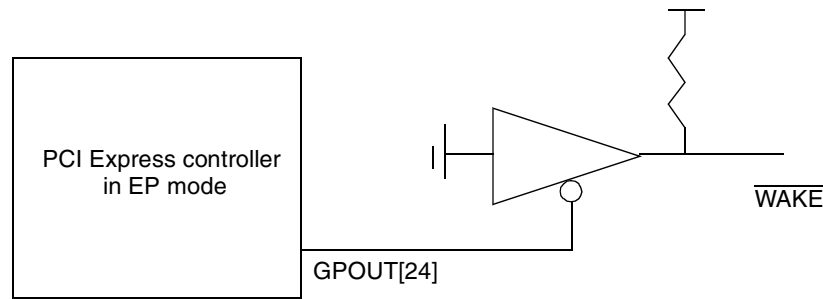


Figure 18-130.  $\overline{\text{WAKE}}$  Generation Example

In RC mode, the  $\overline{\text{WAKE}}$  signal from the EP device can be connected to one of the external interrupt inputs to service the  $\overline{\text{WAKE}}$  request.

### 18.4.5 Hot Reset

When a hot reset condition occurs, the controller (in both RC and EP mode) will initiate a clean-up of all outstanding transactions and returns to an idle state. All configuration register bits that are non-sticky will be reset. Link training will take place subsequently. The device is permitted to generate a hot reset condition on the bus when it is configured as an RC device by setting the “Secondary Bus Reset” bit in the Bridge Control Register in the configuration space. As an EP device, it is not permitted to generate a hot reset condition; it can only detect a hot reset condition and initiate the clean-up procedure appropriately.

## 18.5 Initialization/Application Information

### 18.5.1 Boot Mode and Inbound Configuration Transactions

In normal boot mode (`cfg_cpu_boot = 1`), the core is allowed to boot and configure the device. During this time, the PCI Express interface will retry all inbound PCI Express configuration transactions. When the core has configured the device to a state where it can accept inbound PCI Express configuration transactions, the boot code should set the `CFG_READY` bit in the `PEX_CFG_READY` register after which inbound PCI Express configuration transactions will be accepted. Refer to [Section 18.3.10.18, “Configuration Ready Register—0x4B0,”](#) for more information about the `CFG_READY` bit.

In boot hold-off mode (`cfg_cpu_boot = 0`), the core is prevented from fetching its first instruction by withholding its internal bus grant. During this time, the PCI Express interface accepts all inbound PCI Express configuration transactions which allows an external host/RC to configure the device. When the external host/RC has configured the device to a state where it can allow the core to fetch code from the boot vector, it sets the `MCMPCR[PORT0_EN]` bit after which the core is granted the internal bus.



## Part IV

# Global Functions and Debug

Part IV defines other global blocks of the MPC8533E. The following chapters are included:

- [Chapter 19, “Global Utilities,”](#) defines the global utilities of the MPC8533E. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- [Chapter 20, “Device Performance Monitor,”](#) describes the performance monitor of the MPC8533E.
- [Chapter 21, “Debug Features and Watchpoint Facility,”](#) describes the debug features and watchpoint monitor of the MPC8533E.



# Chapter 19

## Global Utilities

This chapter describes the global utilities of the MPC8533E. It provides signal descriptions, register descriptions, and a functional description of these utilities.

### 19.1 Overview

The global utilities block controls power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal configuration, alternate function selection for multiplexed signals, and clock control.

### 19.2 Global Utilities Features

This section provides an overview of global utilities features.

#### 19.2.1 Power Management and Block Disables

The following features affect the device's overall power consumption:

- Dynamic power management mode
- Software-controlled power management (doze, nap, sleep)
- Externally controlled power management (doze, sleep)
- Static power management (I/O block disables)

#### 19.2.2 Accessing Current POR Configuration Settings

The POR configuration values of all device parameters sampled from pins at reset are available through memory-mapped registers in the global utilities block.

#### 19.2.3 Interrupt and Local Bus Signal Multiplexing

IRQ[9:11] and  $\overline{\text{LCS}}$ [5:7] serve multiple functions that can be selected by configuration registers in the global utilities block.

#### 19.2.4 Clock Control

The global utilities block also selects the internal clock signal driven on CLK\_OUT.

## 19.3 External Signal Description

The following subsections provide information about signals that serve as global utilities.

### 19.3.1 Signals Overview

Table 19-1 summarizes the external signals used by the global utilities block.

**Table 19-1. External Signal Summary**

Signal Name	I/O	Description	Reference (Section/page)
ASLEEP	O	Signals that the device has reached a sleep state.	19.5.1.5.3/19-28
$\overline{\text{CKSTP\_IN}}$	I	Checkstop input	Table 19-2 on page 19-2
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop output.	Table 19-2 on page 19-2
CLK_OUT	O	Clock out. Selected by CLKOCR values.	19.4.1.23/19-23
GPOUT[0:7]	O	General-purpose output.	19.4.1.9/19-12

### 19.3.2 Detailed Signal Descriptions

Table 19-2 describes the global utilities block signals in detail.

**Table 19-2. Detailed Signal Descriptions**

Signal	I/O	Description
ASLEEP	O	Asleep. See <a href="#">Section 19.5.1.5.3, "Sleep Mode."</a> After negation of $\overline{\text{HRESET}}$ , ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.
		<b>State Meaning</b> Asserted—Indicates that the device is either still in its power-on reset sequence or it has reached a sleep state after a power-down command is issued by software. Negated—The device is not in sleep mode. (It has either awakened from a power-down state, or has completed the POR sequence.)
		<b>Timing</b> Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous.
$\overline{\text{CKSTP\_IN}}$	I	Checkstop in
		<b>State Meaning</b> Asserted—Indicates that the e500 core must enter a hard stop condition. All e500 clocks are turned off. $\overline{\text{CKSTP\_OUT}}$ is asserted. The rest of MPC8533E device logic, including memory controllers, internal memories and registers, and I/O interfaces, remains functional. Negated—Indicates that normal operation should proceed.
		<b>Timing</b> Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the MPC8533E is reset with assertion of $\overline{\text{HRESET}}$ .

Table 19-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
CKSTP_OUT	O	Checkstop out	
		<b>State Meaning</b>	Asserted—Indicates that the e500 core of the MPC8533E is in a checkstop state. The rest of the MPC8533E logic remains functional unless. Negated—Indicates normal operation. After $\overline{\text{CKSTP\_OUT}}$ has been asserted, it is negated after the next negation (low-to-high transition) of $\overline{\text{HRESET}}$ .
		<b>Timing</b>	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the device has been reset with a hard reset.
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR (see <a href="#">Section 19.4.1.23, “Clock Out Control Register (CLKOCR)”</a> ).	
		<b>State Meaning</b>	Asserted—If CLKOCR[ENB] = 1, clock signal selected by CLKOCR[CLK_SEL] is driven. High impedance—If CLKOCR[ENB] = 0.
		<b>Timing</b>	Assertion/Negation—Depends on the value of CLKOCR[CLK_SEL].
GPOUT[0:7]	O	General-Purpose Output. See <a href="#">Table 19.4.1.9 on page 19-12</a> for details.	

## 19.4 Memory Map/Register Definition

[Table 19-3](#) summarizes the global utilities registers and their addresses.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 19-3. Global Utilities Block Register Summary

Offset	Register	Access	Reset	Section/page
<b>Power-On Reset Configuration Values</b>				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00nn_n0nn	<a href="#">19.4.1.1/19-4</a>
0xE_0004	PORBMSR—POR boot mode status register	R	0xn000_0000	<a href="#">19.4.1.2/19-6</a>
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	Mixed	0x000n_007F	<a href="#">19.4.1.3/19-7</a>
0xE_000C	PORDEVSR—POR I/O device status register	R	see ref.	<a href="#">19.4.1.4/19-8</a>
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	<a href="#">19.4.1.5/19-10</a>
0xE_0014	PORDEVSR2—POR I/O device status register 2	R	see ref.	<a href="#">19.4.1.6/19-10</a>
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	<a href="#">19.4.1.7/19-11</a>
<b>Signal Multiplexing and GPIO Controls</b>				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	<a href="#">19.4.1.8/19-11</a>

Table 19-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/page
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	<a href="#">19.4.1.9/19-12</a>
0xE_0050	GPINDR—General-purpose input data register	R	0xnnnn_0000	<a href="#">19.4.1.10/19-13</a>
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	<a href="#">19.4.1.11/19-13</a>
<b>Device Disables</b>				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	<a href="#">19.4.1.12/19-14</a>
<b>Power Management Registers</b>				
0xE_0080	POWMGTCSR—Power management status and control register	Mixed	0x0000_0000	<a href="#">19.4.1.13/19-16</a>
<b>Interrupt and Reset Status and Control Registers</b>				
0xE_0090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	<a href="#">19.4.1.14/19-17</a>
0xE_0094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	<a href="#">19.4.1.15/19-18</a>
<b>Version Registers</b>				
0xE_00A0	PVR—Processor version register	R	e500 processor version	<a href="#">19.4.1.16/19-18</a>
0xE_00A4	SVR—System version register	R	MPC8533E system version	<a href="#">19.4.1.17/19-19</a>
<b>Status and Control Registers</b>				
0xE_00B0	RSTCR—Reset control register	R/W	0x0000_0000	<a href="#">19.4.1.18/19-20</a>
0xE_00C0	LBCVSELCR—LBC voltage select control register	R/W	0x0000_0000	<a href="#">19.4.1.19/19-20</a>
0xE_0B20	DDRCSR—DDR configuration status register	R	0x0000_0000	<a href="#">19.4.1.20/19-21</a>
0xE_0B24	DDRCDR—DDR control driver register	R/W	0x0000_0000	<a href="#">19.4.1.21/19-21</a>
0xE_0B28	DDRCLKDR—DDR clock disable register	R/W	0x0000_0000	<a href="#">19.4.1.22/19-22</a>
0xE_0E00	CLKOCR—Clock out control register	R/W	0x0000_0000	<a href="#">19.4.1.23/19-23</a>
0xE_0F04	SRDS1CR1—SerDes 1 control register 1	R/W	0x1100_4418	<a href="#">19.4.1.24/19-24</a>
0xE_0F10	SRDS2CR1—SerDes 2 control register 1	R/W	0x1100_4418	<a href="#">19.4.1.25/19-25</a>

## 19.4.1 Register Descriptions

This section describes the global utilities registers in detail.

### 19.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in [Figure 19-1](#), contains the settings for the PLL ratios as set by the `cfg_sys_pll[0:3]`, `cfg_core_pll[0:1]`, and `cfg_pci_clk` POR configuration pins. See [Section 4.4.3.1, “System PLL Ratio,”](#) and [Section 4.4.3.2, “e500 Core PLL Ratio,”](#) for more information.



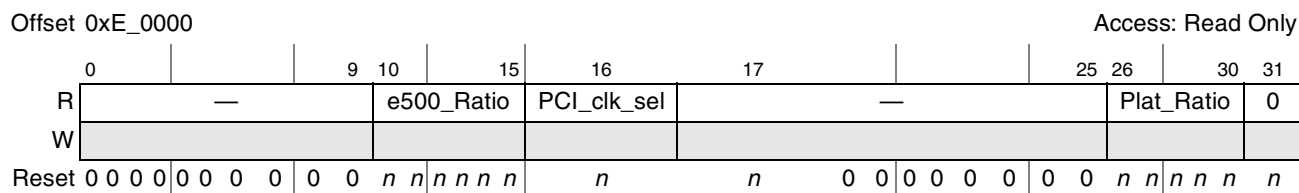


Figure 19-1. POR PLL Status Register (PORPLLSR)

Table 19-4 describes the bit settings of PORPLLSR.

Table 19-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	e500_Ratio	Clock ratio between the e500 core and the CCB clock. The 3 lsbs of this field correspond to the values on <code>cfg_core_pll[0:2]</code> at the negation of $\overline{\text{HRESET}}$ . Patterns not shown are reserved. 000010 1:1 <span style="float: right;">000110 3:1</span> 000011 3:2 <span style="float: right;">000111 7:2</span> 000100 2:1 <span style="float: right;">001000 4:1</span> 000101 5:2 <span style="float: right;">001001 9:2</span>
16	PCI_clk_sel	Clock used for PCI. This bit corresponds to the values on <code>cfg_pci_clk_sel</code> at the negation of $\overline{\text{HRESET}}$ : 0 PCI runs off of PCI_CLK 1 PCI runs off of SYSCLK <sup>1</sup>
17–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the CCB (platform) clock and SYSCLK. The 4 lsbs correspond to the values on <code>cfg_sys_pll[0:3]</code> at the negation of $\overline{\text{HRESET}}$ . Patterns not shown are reserved. 00010 2:1 <sup>1</sup> <span style="float: right;">01000 8:1</span> 00011 3:1 <span style="float: right;">01001 9:1</span> 00100 4:1 <span style="float: right;">01010 10:1</span> 00101 5:1 <span style="float: right;">01100 12:1</span> 00110 6:1 <span style="float: right;">10000 16:1</span> <span style="float: right;">11101 20:1</span>
31	—	Reserved

<sup>1</sup> 2:1 CCB to SYSCLK clock ratio may not be selected when `PCI_clk_sel` are configured to run off of SYSCLK.

### 19.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 19-2, reports setting of the POR configuration pins that control the boot mode settings (described in Section 4.4.3.4, “Boot ROM Location,” Section 4.4.3.7, “CPU Boot Configuration,” and Section 4.4.3.8, “Boot Sequencer Configuration”) and the default settings of PCI/PCI Expresses host/agent mode (described in Section 4.4.3.5, “Host/Agent Configuration”).

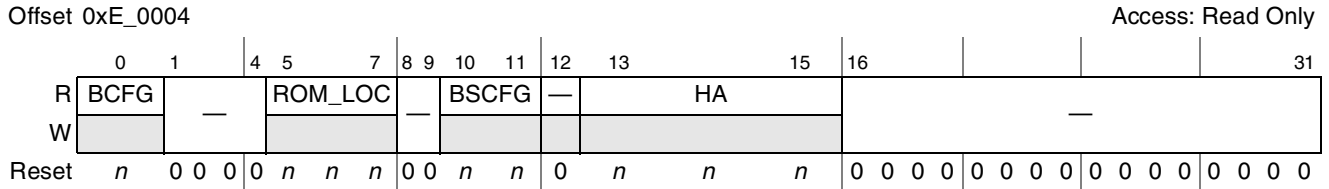


Figure 19-2. POR Boot Mode Status Register (PORBMSR)

For more information about the PCI configurations, see Section 17.3.2.19, “PCI Bus Function Register (PBFR).” Figure 19-5 describes the bit settings of the PORBMSR.

Table 19-5. PORBMSR Field Descriptions

Bits	Name	Description
0	BCFG	CPU boot configuration 0 The CPU is prevented from booting until configuration by an external master is complete. 1 The CPU is allowed to start fetching boot code.
1–4	—	Reserved
5–7	ROM_LOC	Location of boot ROM. This field reflects the values on <code>cfg_rom_loc[0:2]</code> at the negation of $\overline{\text{HRESET}}$ . 000 PCI 001 DDR SDRAM 010 PCI Express 2 011 PCI Express 3 100 PCI Express 1 101 Local bus GPCM: 8-bit 110 Local bus GPCM:16-bit 111 Local bus GPCM: 32-bit
8–9	—	Reserved
10–11	BSCFG	Boot sequencer configuration 00 Reserved 01 Boot sequencer enabled with normal I <sup>2</sup> C addressing 10 Boot sequencer enabled with extended I <sup>2</sup> C addressing 11 Boot sequencer disabled
12	—	Reserved

Table 19-5. PORBMSR Field Descriptions (continued)

Bits	Name	Description
13–15	HA	Host/agent mode configuration. When the MPC8533E is an agent on an interface, it is prevented from mastering transactions on that interface until the external host configures the interface appropriately. 000 Reserved 001 PCI Express 3 endpoint 010 Reserved 011 PCI Express 2 endpoint 100 Reserved 101 PCI Express 1 endpoint 110 PCI Agent 111 Host mode/Root complex
16–31	—	Reserved

### 19.4.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

PORIMPSCR, shown in Figure 19-3, contains the current I/O driver impedances for local bus and PCI interfaces.

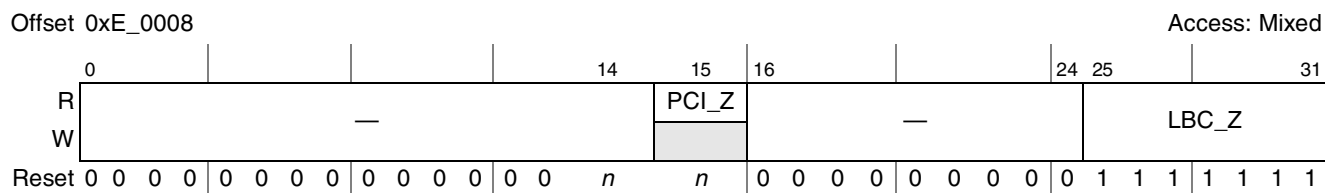


Figure 19-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

The I/O impedance of local bus signals (including the local bus clock) is controlled through this register. The I/O impedance of PCI signals is controlled by POR configuration pins (described in Section 4.4.3.16, “PCI I/O Impedance”). The *MPC8533E Integrated Processor Hardware Specification* provides exact I/O impedances.

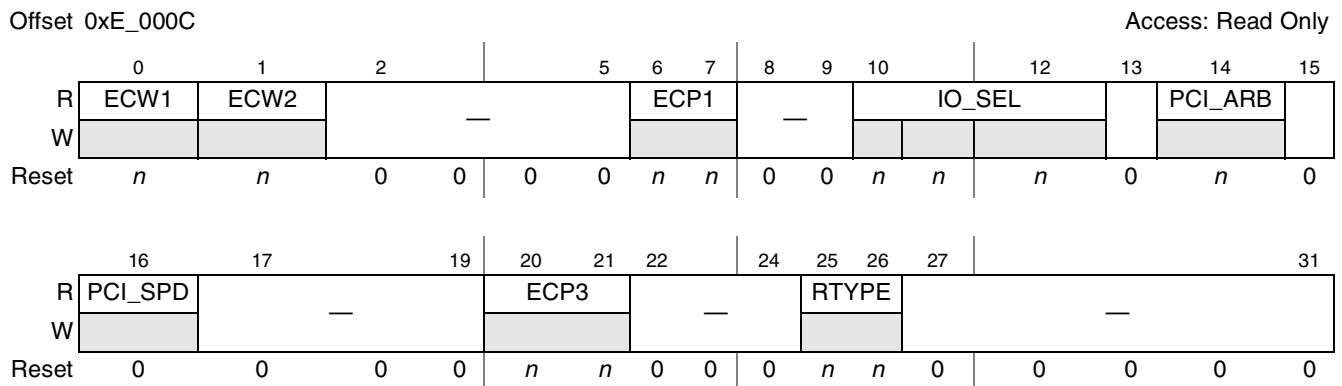
Table 19-6 describes PORIMPSCR fields.

Table 19-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	PCI_Z	PCI I/O impedance 0 Low impedance 1 High impedance
16–24	—	Reserved
25–31	LBC_Z	I/O impedance for these local bus signals: LAD[0:31], LDP[0:3], LA[27:31], $\overline{\text{LCS}}$ [1:2], $\overline{\text{LWE}}$ [0:3], LGPL[0:5], LCKE, LCLK[0:2] Note: Other signals (for example, LALE, $\overline{\text{LCS}}$ [0], $\overline{\text{LCS}}$ [3:7]) use a fixed high I/O impedance 111111 High impedance else Low impedance

### 19.4.1.4 POR Device Status Register (PORDEVSR)

Shown in [Figure 19-4](#), PORDEVSR reports other POR settings for I/O devices as described in [Section 4.4.3.10](#), “eTSEC1 Width,” [Section 4.4.3.11](#), “eTSEC3 Width,” and [Section 4.4.3.17](#), “PCI Arbiter Configuration.”



**Figure 19-4. POR Device Status Register (PORDEVSR)**

[Table 19-7](#) describes the bit settings of PORDEVSR.

**Table 19-7. PORDEVSR Field Descriptions**

Bits	Name	Description
0	ECW1	eTSEC1 controller width (See <a href="#">Section 4.4.3.10</a> , “eTSEC1 Width.”) 0 Reduced interfaces for eTSEC1(RGMII, RTBI, or 8-bit FIFO on eTSEC1) 1 Full interfaces for eTSEC1(MII, GMII, TBI, or 16-bit FIFO on eTSEC1)
1	ECW2	eTSEC3 controller width (See <a href="#">Section 4.4.3.11</a> , “eTSEC3 Width.”) 0 Reduced interfaces for eTSEC3 (RGMII, RTBI) 1 Full interface for eTSEC3 (MII, GMII, TBI) note: FIFO mode on eTSEC3 is always 8-bits regardless of ECW2.
2	—	Reserved
3	—	Reserved
4	—	Reserved
5	—	Reserved
6–7	ECP1	eTSEC1 controller protocol (See <a href="#">Section 4.4.3.12</a> , “eTSEC1 Protocol.”) 00 The eTSEC1 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC1 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode).
8–9	—	Reserved

Table 19-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
10–12	IO_SEL	I/O port selection mode (See <a href="#">Section 4.4.3.6, “I/O Port Selection.”</a> ) 000 All PCI Express ports powered down 001 Reserved 010 PCI Express port 1 active, PCI Express port 2 and 3 powered down 011 Reserved 100 PCI Express port 1 and 2 active, PCI Express port 3 powered down 101 Reserved 110 All PCI Express active 111 Reserved  PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane [0:3] -> SD1_TX[0:3] PCI Express 2: RX lane[0:3] -> SD1_RX[4:7] TX lane [0:3] -> SD1_TX[4:7] PCI Express 3: RX lane[0] -> SD2_RX[0] TX lane [0] -> SD2_TX[0]
13	—	Reserved
14	PCI_ARB	PCI arbiter enable (See <a href="#">Section 4.4.3.17, “PCI Arbiter Configuration.”</a> ) 0 PCI arbiter is disabled 1 PCI arbiter is enabled
15	—	Reserved
16	PCI_SPD	PCI speed (See <a href="#">Section 4.4.3.15, “PCI Speed Configuration.”</a> ) 0 PCI set for low speed operation—PCI below 33MHz 1 PCI set for normal speed operation—PCI at or above 33MHz
17–19	—	Reserved
20–21	ECP3	eTSEC3 controller protocol (See <a href="#">Section 4.4.3.13, “eTSEC3 Protocol.”</a> ) 00 The eTSEC3 controller operates using the 8-bit FIFO protocol. 01 The eTSEC3 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC3 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC3 controller operates using the TBI protocol (or RTBI if configured in reduced mode).
22–24	—	Reserved
25–26	RTYPE	DRAM Type (See <a href="#">Section 4.4.3.9, “DDR SDRAM Type.”</a> ) 00 Low power DDR I 01 DDR I 10 Reserved 11 DDR II
27–31	—	Reserved

### 19.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 19-5, holds debug mode settings from the POR configuration pins as described in Section 4.4.3.18, “Memory Debug Configuration,” and Section 4.4.3.19, “DDR Debug Configuration.”

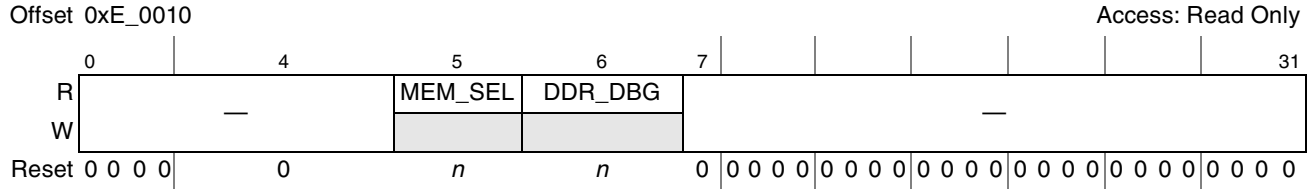


Figure 19-5. POR Debug Mode Status Register (PORDBGMSR)

Table 19-8 describes the bit settings of PORDBGMSR.

Table 19-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	MEM_SEL	Memory select. Indicates which controller is driving MSRCID[0:4] and MDVAL. 0 Local bus controller is driving debug information 1 DDR SDRAM controller is driving debug information
6	DDR_DBG	DDR debug configuration 0 SourceID and data valid information is being driven on ECC pins of DDR SDRAM interface 1 Normal mode. ECC information is being driven on ECC pins of DDR SDRAM interface
7–31	—	Reserved

### 19.4.1.6 POR Device Status Register 2 (PORDEVSR2)

Figure 19-6 describes the bit settings of PORDEVSR2.

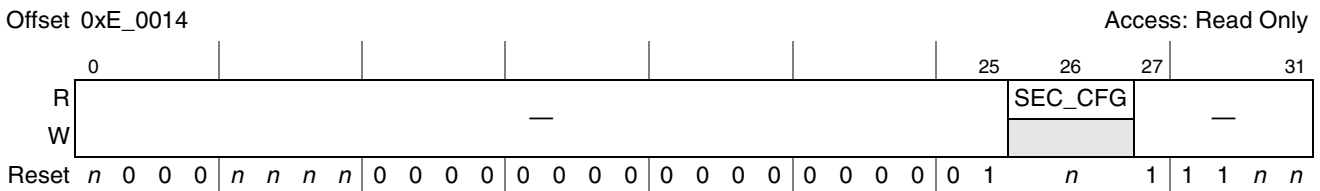


Figure 19-6. POR Device Status Register 2 (PORDEVSR2)

Table 19-9 describes the bit settings of PORDEVSR2.

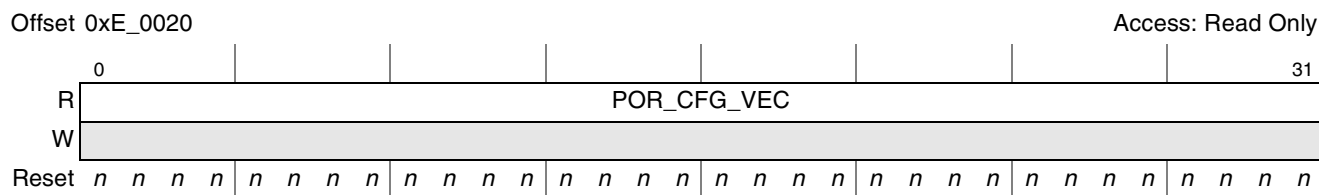
**Table 19-9. PORDEVSR2 Field Descriptions**

Bits	Name	Description
0–25	—	Reserved
26	SEC_CFG	SEC frequency ratio configuration (See <a href="#">Section 4.4.3.3, “SEC Frequency Ratio Configuration.”</a> ) 0 Enables the SEC in 2:1 (CCB CLK: SEC CLK) mode with respect to e500 core complex bus (CCB) clock 1 Enables the SEC in 3:1 (CCB CLK: SEC CLK) mode with respect to e500 core complex bus (CCB) clock
27–31	—	Reserved

### 19.4.1.7 General-Purpose POR Configuration Register (GPPORCR)

GPPORCR stores the value sampled from the local bus address/data signals, LAD[0:31], during POR, as described in [Section 4.4.3.20, “General-Purpose POR Configuration.”](#) Software can use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

GPPORCR is shown in [Figure 19-7](#).



**Figure 19-7. POR Configuration Register (GPPORCR)**

Table 19-10 describes the bit settings of GPPORCR.

**Table 19-10. GPPORCR Field Descriptions**

Bits	Name	Description
0–31	POR_CFG_VEC	General-purpose POR configuration vector sampled from local bus address/data signals at the negation of HRESET. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

### 19.4.1.8 General-Purpose I/O Control Register (GPIOCR)

Shown in [Figure 19-8](#), GPIOCR contains the enable bits for each group of pins that may be used for general-purpose I/O. Note that when these signals are enabled as general-purpose I/O signals, they are read and written through GPINDR and GPOUTDR described in [Section 19.4.1.10, “General-Purpose Input Data Register \(GPINDR\),”](#) and [Section 19.4.1.9, “General-Purpose Output Data Register \(GPOUTDR\).”](#) [Section 19.5.2, “General-Purpose I/O Signals,”](#) describes the use of general-purpose I/O signals.

Offset 0xE\_0030

Access: Read/Write

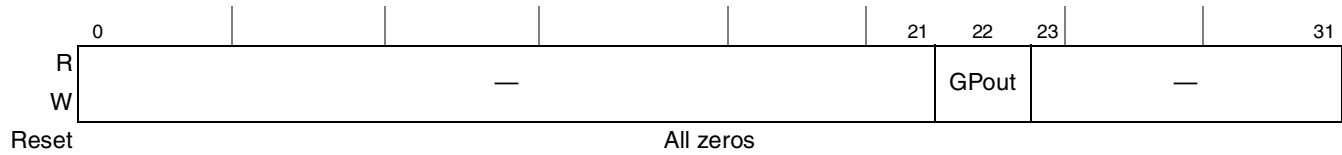


Figure 19-8. General-Purpose I/O Control Register (GPIOCR)

Table 19-11 describes the bit settings of GPIOCR.

Table 19-11. GPIOCR Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	GPout	Enables GPOUT[0:7] for use as general-purpose output.
23–31	—	Reserved

### 19.4.1.9 General-Purpose Output Data Register (GPOUTDR)

Writes to GPOUTDR affect only pins enabled as general-purpose outputs. Reads return valid data only for bits corresponding to pins enabled as general-purpose outputs. GPOUTDR may be accessed using single byte writes (using big-endian addressing) so that writes to one byte do not affect outputs controlled by others.

Offset 0xE\_0040

Access: Read/Write

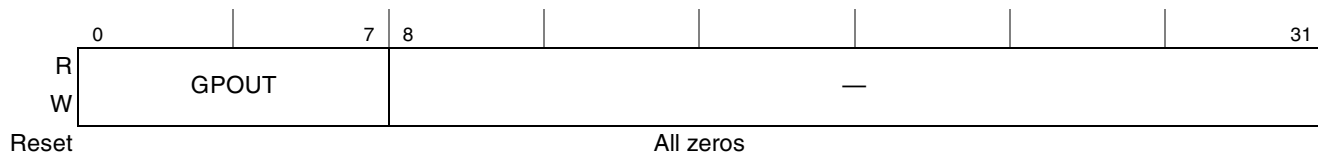


Figure 19-9. General-Purpose Output Data Register (GPOUTDR)

Table 19-12 describes the fields of GPOUTDR.

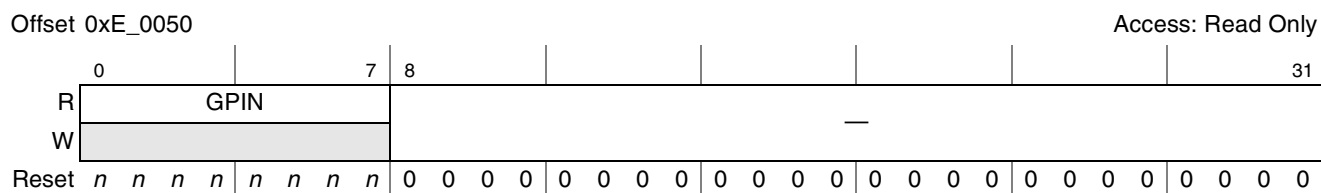
Table 19-12. GPOUTDR Field Descriptions

Bits	Name	Description
		General-purpose output data. When the corresponding signals are configured to be general-purpose output signals, the values of the bits of GPOUT are driven onto those pins.
0–7	GPOUT	General-purpose output data. The values of the bits of field GPOUTDR[0:7] are driven onto pins GPOUT[0:7] as follows: GPOUTDR[0] ↔ GPOUT[0] GPOUTDR[1] ↔ GPOUT[1] GPOUTDR[2] ↔ GPOUT[2] GPOUTDR[3] ↔ GPOUT[3] GPOUTDR[4] ↔ GPOUT[4] GPOUTDR[5] ↔ GPOUT[5] GPOUTDR[6] ↔ GPOUT[6] GPOUTDR[7] ↔ GPOUT[7]
8–31	—	Reserved, should be cleared



### 19.4.1.10 General-Purpose Input Data Register (GPINDR)

GPINDR, shown in [Figure 19-10](#), contains the data currently sampled as general-purpose input on GPIN pins.



**Figure 19-10. General-Purpose Input Data Register (GPINDR)**

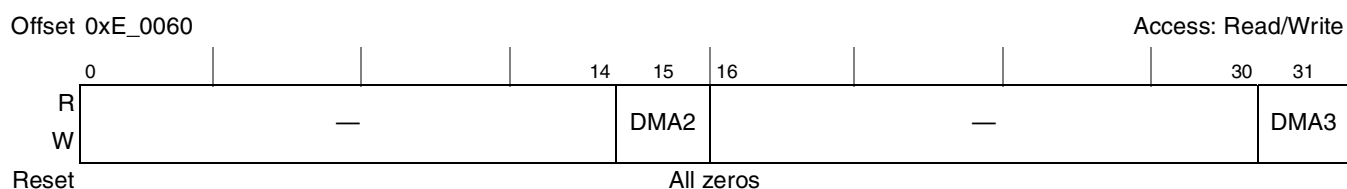
[Table 19-12](#) describes the fields of GPINDR.

**Table 19-13. GPINDR Field Descriptions**

Bits	Name	Description
0–7	GPIN	General-purpose input data. The values of the bits of field GPINDR[0:7] are reflected onto pins GPIN[0:7] as follows: GPINDR[0] ↔ GPIN[0] GPINDR[1] ↔ GPIN[1] GPINDR[2] ↔ GPIN[2] GPINDR[3] ↔ GPIN[3] GPINDR[4] ↔ GPIN[4] GPINDR[5] ↔ GPIN[5] GPINDR[6] ↔ GPIN[6] GPINDR[7] ↔ GPIN[7]
8–31	—	Reserved

### 19.4.1.11 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in [Figure 19-11](#), PMUXCR contains bits that enable DMA channels 2 and 3 which exist as alternate functions on local bus chip select pins  $\overline{\text{LCS}}[5:7]$ , and interrupt input pins IRQ[9:11], respectively. Specifically, DMA request, acknowledge, and done signals comprise the secondary functions for the associated IRQ and local bus chip select signals.



**Figure 19-11. Alternate Function Pin Multiplex Control Register (PMUXCR)**

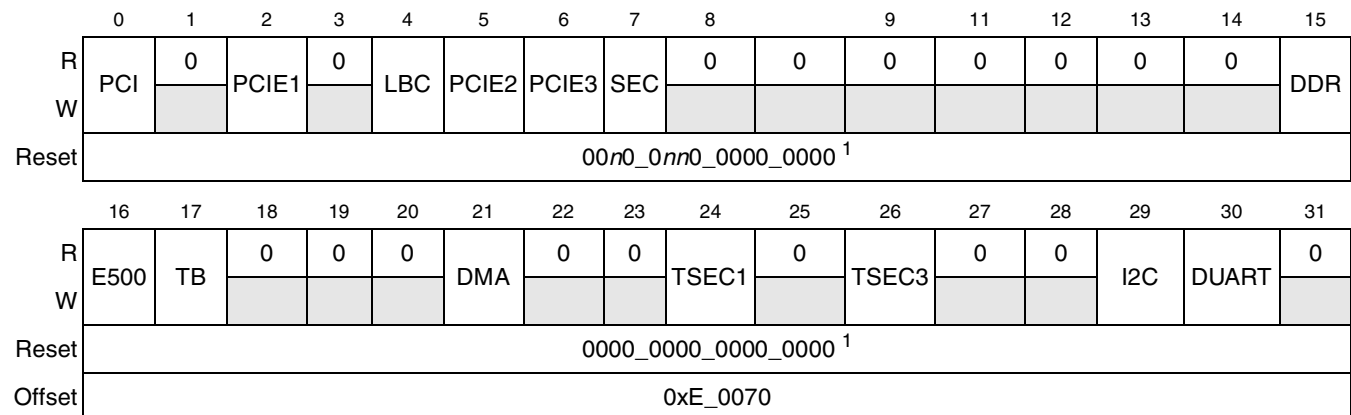
Table 19-14 describes the bit settings of PMUXCR.

**Table 19-14. PMUXCR Field Descriptions**

Bits	Name	Description
0–14	—	Reserved
15	DMA2	Enables DMA channel 2 signals. 0 DMA channel 2 is not exposed to pins; the pins retain their primary function as local bus chip selects 1 DMA channel 2 is exposed to pins as follows: LCS5 functions as <u>DMA_DREQ2</u> LCS6 functions as <u>DMA_DACK2</u> LCS7 functions as <u>DMA_DDONE2</u>
16–30	—	Reserved
31	DMA3	Enables DMA channel 3 signals. 0 DMA channel 3 is not exposed to pins; the pins retain their primary function as interrupt requests. 1 DMA channel 3 is exposed to pins as follows: IRQ9 functions as <u>DMA_DREQ3</u> IRQ10 functions as <u>DMA_DACK3</u> IRQ11 functions as <u>DMA_DDONE3</u>

### 19.4.1.12 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 19-12, contains disable bits for various MPC8533E functional blocks.



**Figure 19-12. Device Disable Register (DEVDISR)**

<sup>1</sup> \* *n* bits depend on the state of the corresponding POR configuration signals at reset.

All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption.

**Table 19-15. DEVDISR Field Descriptions**

Bits	Name	Description
0	PCI	PCI controller disable 0 PCI controller enable 1 PCI controller disable
1	—	Reserved

Table 19-15. DEVDISR Field Descriptions (continued)

Bits	Name	Description
2	PCIE1	PCI Express 1 controller disable 0 PCI Express 1 controller enable 1 PCI Express 1 controller disable
3	—	Reserved
4	LBC	Local bus controller disable 0 Local bus controller enable 1 Local bus controller disable
5	PCIE2	PCI Express 2 controller disable 0 PCI Express 2 controller enable 1 PCI Express 2 controller disable
6	PCIE3	PCI Express 3 controller disable 0 PCI Express 3 controller enable 1 PCI Express 3 controller disable
7	SEC	Security disable 0 Security enable 1 Security disable
8–14	—	Reserved
15	DDR	DDR SDRAM controller disable 0 DDR SDRAM controller enable 1 DDR SDRAM controller disable
16	E500	e500 core disable 0 e500 core enable 1 e500 core disable. Places the core in the core_stopped state in which it does not respond to interrupts. Equivalent to nap mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core including the timer facilities. For more information, see <a href="#">Section 19.5.1.4, “Shutting Down Unused Blocks.”</a>
17	TB	Time base (timer facilities) of the e500 core disable 0 Timer facilities enabled 1 Timer facilities disabled.
18–20	—	Reserved
21	DMA	DMA controller disabled 0 DMA controller enabled 1 DMA controller disabled
22–23	—	Reserved
24	TSEC1	Three-speed Ethernet controller 1 disable 0 eTSEC1 enabled 1 eTSEC1 disabled
25	—	Reserved
26	TSEC3	Three-speed Ethernet controller 3 disable 0 eTSEC3 enabled 1 eTSEC3 disabled
27–28	—	Reserved

Table 19-15. DEVDISR Field Descriptions (continued)

Bits	Name	Description
29	I2C	I <sup>2</sup> C controllers disabled 0 I <sup>2</sup> C controllers enabled 1 I <sup>2</sup> C controllers disabled
30	DUART	Dual UART controller disabled 0 DUART enabled 1 DUART disabled
31	—	Reserved

### 19.4.1.13 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 19-13, POWMGTCR contains bits for placing the MPC8533E into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 19.5.1.8.2, “Interrupts and Power Management Controlled by POWMGTCR,” for more information.

Offset 0xE\_0080

Access: Mixed

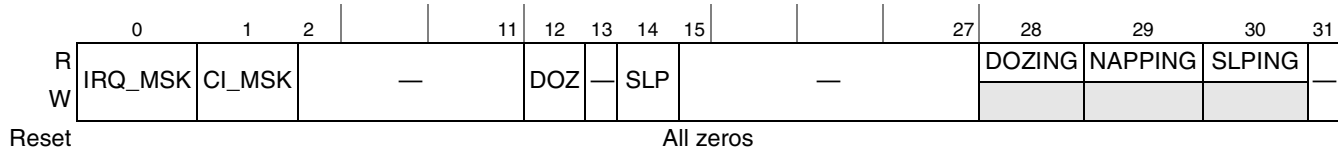


Figure 19-13. Power Management Control and Status Register (POWMGTCSR)

Table 19-16 describes the bit settings of POWMGTCR.

Table 19-16. POWMGTCR Field Descriptions

Bits	Name	Description
0	IRQ_MSK	Interrupt input mask 0 Interrupts cause the device to wake up from a low-power state. 1 Interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of an interrupt request.
1	CI_MSK	Critical interrupt input mask 0 Critical interrupts cause the device to wake up from a low power state. 1 Critical interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of a critical interrupt.
2–11	—	Reserved
12	DOZ	Doze mode. 0 No request to put device in doze mode. Note that this bit is automatically cleared on MCP, UDE, SRESET, <i>core_tbit</i> (from the core) and also <i>int</i> and <i>cin</i> if not masked. 1 Device is to be placed in doze mode. Instruction fetching is halted in the e500 core. Note that this bit is logically ORed with HID0[DOZE].
13	—	Reserved

Table 19-16. POWMGTCR Field Descriptions (continued)

Bits	Name	Description
14	SLP	Sleep mode 0 No request to put device in sleep mode. 1 Device is to be placed in sleep mode. Instruction fetching is halted, snooping of L1 caches is disabled, and most functional blocks are shut down in both the e500 core and the system logic.
15–27	—	Reserved
28	DOZING	Doze status 0 Device is not in doze mode. 1 The MPC8533E is in doze mode because POWMGTCR[DOZ] is set or because HID0[DOZE] and MSR[WE] (in the e500 core) are set. The core has halted instruction fetching, but all other functional blocks in the core and device are running.
29	NAPPING	Nap status 0 Device is not in nap mode. 1 The MPC8533E is in nap mode because HID0[NAP] and MSR[WE] are set. The core has halted instruction fetching, snooping of the L1 caches is disabled, and all of the core's functional units except the timer facilities are shut down. All functional blocks in the device are running.
30	SLPING	Sleep status 0 Device is not attempting to reach sleep mode. 1 The device is attempting to SLEEP because POWMGTCR[SLP] is set or because HID0[SLEEP] and MSR[WE] (in the e500 core) are set. Most functional blocks in the core and device are shut down or are attempting to shut down.
31	—	Reserved. Should be cleared.

#### 19.4.1.14 Machine Check Summary Register (MCPSUMR)

Shown in Figure 19-14, MCPSUMR contains bits summarizing some of the sources of a pending machine check interrupt. All MCPSUMR bits function as write-1-to-clear.

#### NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

Note that other conditions can cause a machine check condition not summarized in MCPSUMR. For example, uncorrectable read errors cause the assertion of *core\_fault\_in*, which may directly cause a machine check (if HID1[RFXE] = 1). If RFXE = 0, the assertion of *core\_fault\_in* does not directly cause a machine check interrupt, but must be handled by the block that generated the error. For more information about RFXE, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

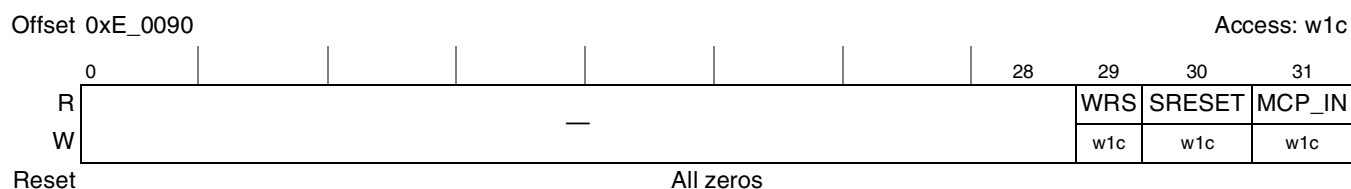


Figure 19-14. Machine Check Summary Register (MCPSUMR)

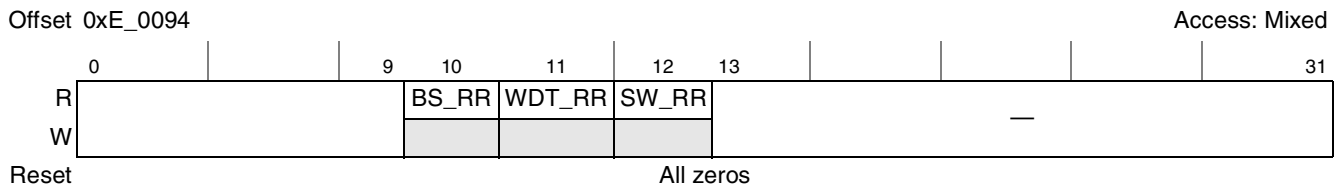
Table 19-17 describes the bit settings of MCPSUMR.

**Table 19-17. MCPSUMR Field Descriptions**

Bits	Name	Description
0–28	—	Reserved
29	WRS	Watchdog timer machine check 0 Machine check exception was not caused by watchdog timer. 1 Machine check was caused by a soft reset condition from the e500 watchdog timer as configured in the core's TSR. Specifically, TSR[WRS] = 01 and a watchdog reset condition occurred.
30	SRESET	Soft reset machine check 0 Machine check exception was not caused by $\overline{\text{SRESET}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{SRESET}}$ input signal.
31	MCP_IN	$\overline{\text{MCP}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{MCP}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP}}$ input signal.

### 19.4.1.15 Reset Request Status and Control Register (RSTRSCR)

Shown in Figure 19-15, the RSTRSCR contains mask bits status for boot sequencer, watchdog timer, and a software settable reset request bit.



**Figure 19-15. Reset Request Status and Control Register (RSTRSCR)**

Table 19-18 describes the bit settings of RSTRSCR.

**Table 19-18. RSTRSCR Field Descriptions**

Bits	Name	Description
0–9	—	Reserved
10	BS_RR	Boot sequence reset request
11	WDT_RR	Watchdog timer reset request
12	SW_RR	Software settable reset request
13–31	—	Reserved

### 19.4.1.16 Processor Version Register (PVR)

Shown in Figure 19-16, the PVR contains the e500 processor version number. It is a memory-mapped copy of the PVR in the e500 core (and is therefore accessible to external devices). Section 5.2, “e500 Processor and System Version Numbers,” lists the complete values for the MPC8533E.

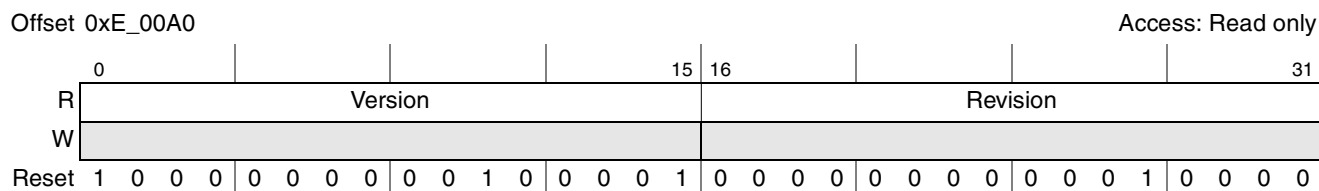


Figure 19-16. Processor Version Register (PVR)

Table 19-19 describes the fields of PVR.

Table 19-19. PVR Field Descriptions

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported. (See <a href="#">Section 5.2, “e500 Processor and System Version Numbers,”</a> for specific values.)
48–63	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level. (See <a href="#">Section 5.2, “e500 Processor and System Version Numbers,”</a> for specific values.)

#### 19.4.1.17 System Version Register (SVR)

Shown in [Figure 19-17](#), the SVR contains the system version number for the MPC8533E implementation. This value can also be read through the SVR SPR of the e500 core. See [Section 6.5.4, “System Version Register \(SVR\).”](#) [Section 5.2, “e500 Processor and System Version Numbers,”](#) lists the complete values for the MPC8533E.

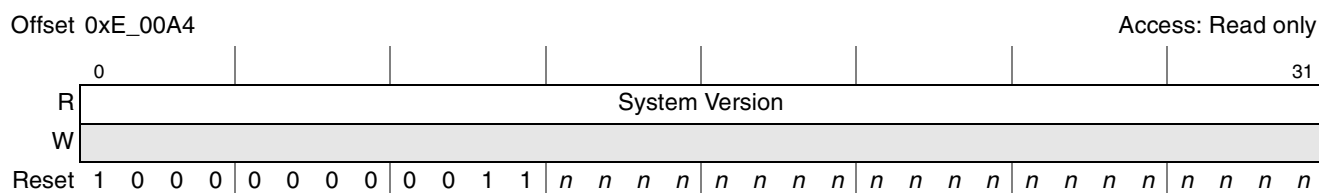


Figure 19-17. System Version Register (SVR)

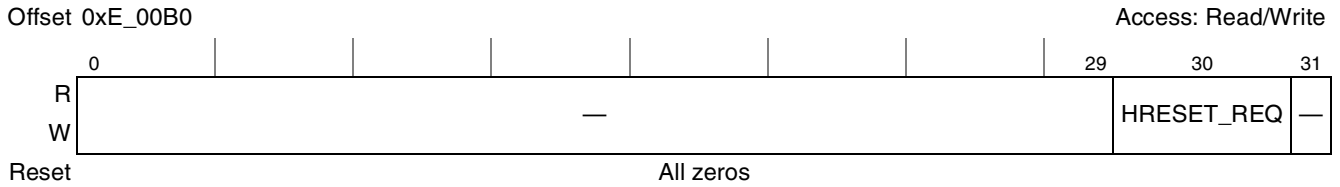
Table 19-20 describes the fields of SVR.

Table 19-20. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version numbers for the MPC8533E system logic 0x803_C_00_10 for MPC8533E with security 0x803_4_00_10 for MPC8533 without security

### 19.4.1.18 Reset Control Register (RSTCR)

Shown in [Figure 19-18](#), the RSTCR contains the reset control bits.



**Figure 19-18. Reset Control Register (RSTCR)**

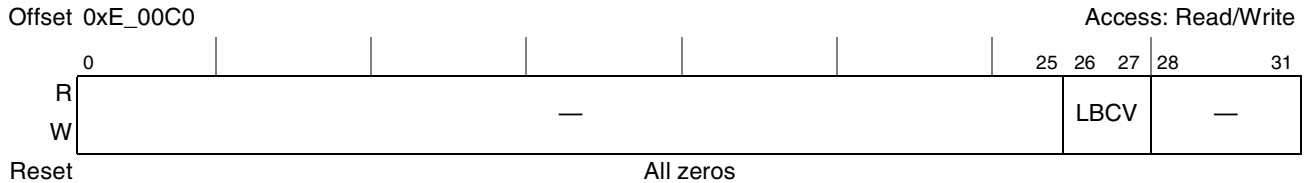
[Table 19-21](#) describes the bit settings of RSTCR.

**Table 19-21. RSTCR Field Descriptions**

Bits	Name	Description
0–29	—	Reserved
30	HRESET_REQ	Hardware reset request
31	—	Reserved

### 19.4.1.19 LBC Voltage Select Control Register (LBCVSELCR)

Shown in [Figure 19-19](#), the LBCVSELCR contains local bus voltage control bits.



**Figure 19-19. LBC Voltage Select Control Register (LBCVSELCR)**

[Table 19-22](#) describes the bit settings of LBCVSELCR.

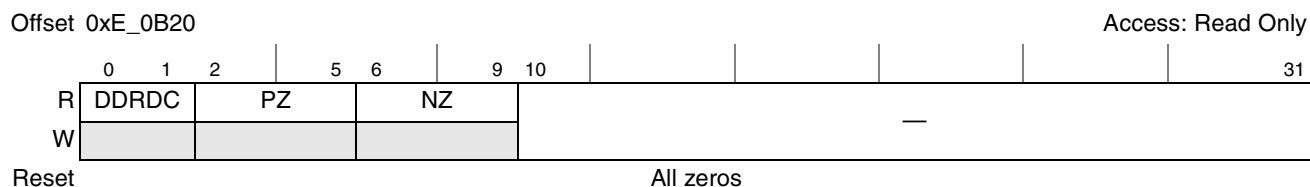
**Table 19-22. LBCVSELCR Field Descriptions**

Bits	Name	Description
0–25	—	Reserved
26–27	LBCV	Selects the I/O voltage for the local bus 00 (default) 3.3V 01 2.5V 10 1.8V 11 reserved
28–31	—	Reserved



### 19.4.1.20 DDR Calibration Status Register (DDRCSSR)

Shown in [Figure 19-20](#), the DDRCSSR contains debug status bits from the DDR SDRAM controller.



**Figure 19-20. DDR Calibration Status Register (DDRCSSR)**

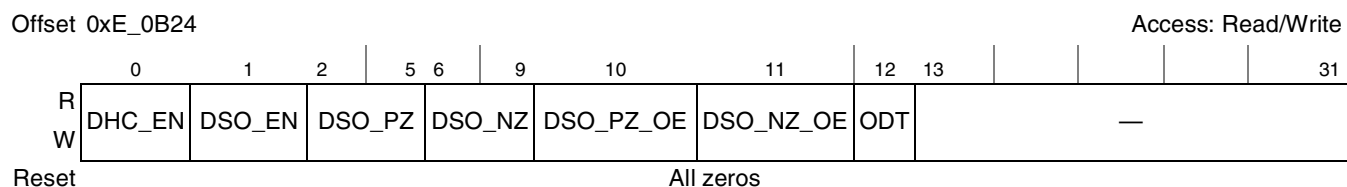
[Table 19-23](#) describes the bit settings of DDRCSSR.

**Table 19-23. DDRCSSR Field Descriptions**

Bits	Name	Description
0–1	DDRDC	DDR driver compensation input value. This field reflects the current state of the MDIC[0:1] driver impedance calibration signals.
2–5	PZ	Current setting of PFET driver impedance (Field values not defined below are reserved.) 0000 Highest impedance; half strength 1000 Higher impedance 1100 Nominal impedance 1110 Lower impedance 1111 Lowest impedance; double strength
6–9	NZ	Current setting of NFET driver impedance (Field values not defined below are reserved.) 0000 Highest impedance; half strength 1000 Higher impedance 1100 Nominal impedance 1110 Lower impedance 1111 Lowest impedance; double strength
10–31	—	Reserved

### 19.4.1.21 DDR Control Driver Register (DDRCDR)

Shown in [Figure 19-21](#), the DDRCDR contains bits that allow control over the I/O drivers of the DDR SDRAM controller.



**Figure 19-21. DDR Control Driver Register (DDRCDR)**

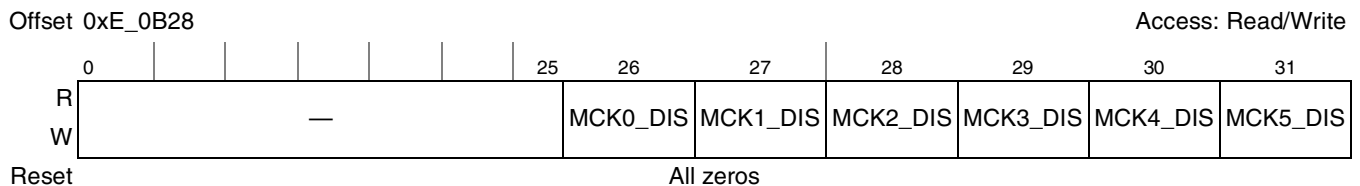
Table 19-24 describes the bit settings of DDRCDR.

**Table 19-24. DDRCDR Field Descriptions**

Bits	Name	Description
0	DHC_EN	DDR driver hardware compensation enable
1	DSO_EN	DDR driver software override enable
2–5	DSO_PZ	DDR driver software p-impedance override
6–9	DSO_NZ	DDR driver software n-impedance override
10	DSO_PZ_OE	DDR driver software p-impedance OE
11	DSO_NZ_OE	DDR driver software n-impedance OE
12	ODT	ODT termination value for IOs 0 ODT termination of 75 ohms 1 ODT termination of 150 ohms
13–31	—	Reserved

### 19.4.1.22 DDR Clock Disable Register (DDRCLKDR)

Shown in Figure 19-22, the DDRCLKDR contains bits that allow disabling the clocks of the DDR SDRAM controller.



**Figure 19-22. DDR Clock Disable Register (DDRCLKDR)**

Table 19-24 describes the bit settings of DDRCLKDR.

**Table 19-25. DDRCLKDR Field Descriptions**

Bits	Name	Description
0–25	—	Reserved
26	MCK0_DIS	DDR clock 0 disable 0 MCK0 is enabled. 1 MCK0 is disabled.
27	MCK1_DIS	DDR clock 1 disable 0 MCK1 is enabled. 1 MCK1 is disabled.
28	MCK2_DIS	DDR clock 2 disable 0 MCK2 is enabled. 1 MCK2 is disabled.
29	MCK3_DIS	DDR clock 3 disable 0 MCK3 is enabled. 1 MCK3 is disabled.

Table 19-25. DDRCLKDR Field Descriptions (continued)

Bits	Name	Description
30	MCK4_DIS	DDR clock 4 disable 0 MCK4 is enabled. 1 MCK4 is disabled.
31	MCK5_DIS	DDR clock 5 disable 0 MCK5 is enabled. 1 MCK5 is disabled.

### 19.4.1.23 Clock Out Control Register (CLKOCR)

Shown in Figure 19-23, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK\_OUT) signal.

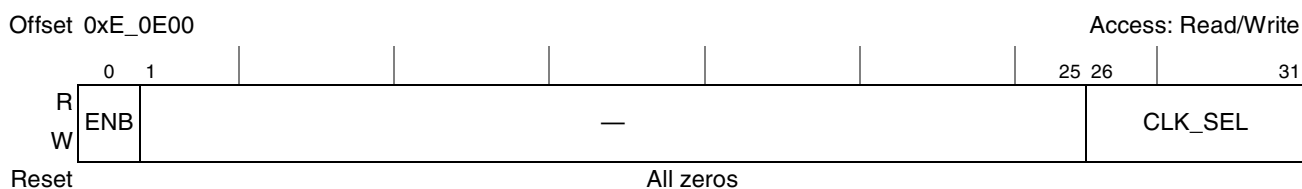


Figure 19-23. Clock Out Control Register (CLKOCR)

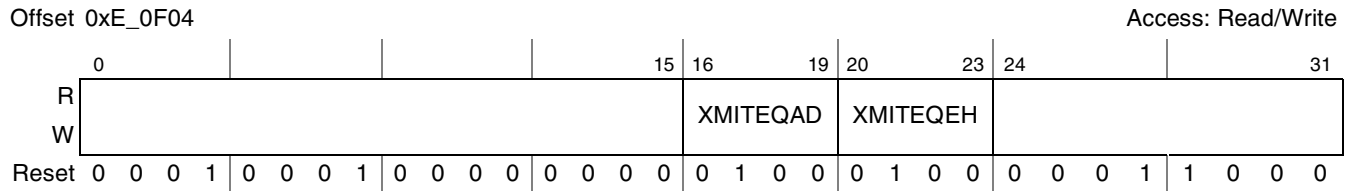
Table 19-26 describes the bit settings of CLKOCR.

Table 19-26. CLKOCR Field Descriptions

Bits	Name	Description
0	ENB	Clock out enable 0 CLK_OUT signal is three-stated 1 CLK_OUT signal is driven according to CLKOCR[CLK_SEL]
1–25	—	Reserved
26–31	CLK_SEL	Clock out select 000000 CCB (platform) clock 01xx1x Reserved 000001 CCB (platform) clock divided by 2 10x000 Reserved 000010 SYSCLK (echoes SYSCLK input) 10x001 Reserved 000011 SYSCLK divided by 2 (demonstrates platform PLL lock) 10x010 PCI bus clock 10x011 PCI bus clock divided by 2 000100 Reserved 10x100 Reserved 000101 Reserved 10x101 Reserved 000110 Reserved 10x110 Reserved 000111 Reserved 10x111 Logic 0 001000 Reserved 11x000 Reserved 001001 Reserved 11x001 Reserved 001010 Reserved 11x010 Reserved 001011 Reserved 11x011 Reserved 001100 Reserved 11x100 Reserved 001101 Reserved 11x101 Reserved 001110 Reserved 11x110 Reserved 001111 Reserved 11x111 Logic 1 01xx0x Reserved

### 19.4.1.24 SerDes 1 Control Register 1 (SRDS1CR1)

Shown in [Figure 19-24](#), SRDS1CR1 contains the control bits used for adjusting the transmit equalization of the SerDes 1 signals. Note that reserved fields with non-zero reset values are for internal use only and their values should be preserved.



**Figure 19-24. SerDes 1 Control Register 1 (SRDS1CR1)**

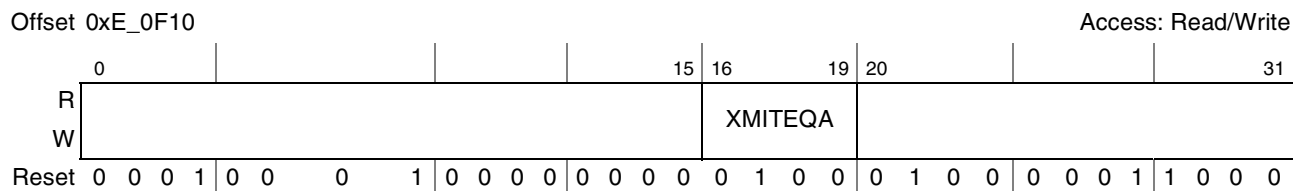
[Table 19-27](#) describes the bit settings of SRDS1CR1.

**Table 19-27. SRDS1CR1 Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–19	XMITEQAD	Transmit equalization selection bus for SerDes 1 lanes a–d (SD1_RX/TX[0:3]) Default value = 4'b0100 MSB (bit 16) is amplitude select: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 17:19) are equalization amplitude: 000 No equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude 111 reserved
20–23	XMITEQEH	Transmit equalization selection bus for SerDes 1 lanes e–h (SD1_RX/TX[4:7]) Default value = 4'b0100 MSB (bit 20) is amplitude select: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 21:23) are equalization amplitude: 000 No equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude 111 reserved
24–31	—	Reserved

### 19.4.1.25 SerDes 2 Control Register 1 (SRDS2CR1)

Shown in [Figure 19-25](#), SRDS2CR1 contains the control bits used for adjusting the transmit equalization of the SerDes 2 signals. Note that reserved fields with non-zero reset values are for internal use only and their values should be preserved.



**Figure 19-25. SerDes 2 Control Register 1 (SRDS2CR1)**

[Table 19-28](#) describes the bit settings of SRDS2CR1.

**Table 19-28. SRDS2CR1 Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–19	XMITEQA	Transmit equalization selection bus for SerDes 2 lane a (SD2_RX0/TX0) Default value = 4'b0100 MSB (bit 16) is the differential peak-to-peak amplitude selection: 0 = V <sub>dd-diff-pk=pk</sub> 1 = 5/6 V <sub>dd-diff-pk=pk</sub> LSBs (bits 17:19) are the equalization amplitude selection: 000 = No equalization 001 = 1.09x relative amplitude 010 = 1.2x relative amplitude 011 = 1.33x relative amplitude 100 = 1.5x relative amplitude 101 = 1.71x relative amplitude 110 = 2.0x relative amplitude 111 = reserved
20–31	—	Reserved

## 19.5 Functional Description

This section describes the global utilities from a functional perspective.

### 19.5.1 Power Management

The MPC8533E has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to individual blocks when they are not needed through a memory-mapped register (DEVDISR). Additionally, software running on the e500 core can access the core's SPRs to put the device into doze, nap, or sleep power down state. Finally, software can access a memory-mapped register (POWMGTCR) in the global utilities block to put the device in the doze or sleep states.

Note that the software that writes to either DEVDISR or POWMGTCR can be running either on the e500 core or on an external master that can write to the MPC8533E memory-mapped registers through the PCI interfaces.

These features are described in further detail in this section.

### 19.5.1.1 Relationship Between Core and Device Power Management States

The MPC8533E has three low-power states: doze, nap, and sleep. The mapping of core and device power management states is shown in Figure 19-26 showing state transitions from the perspective of the e500 core.

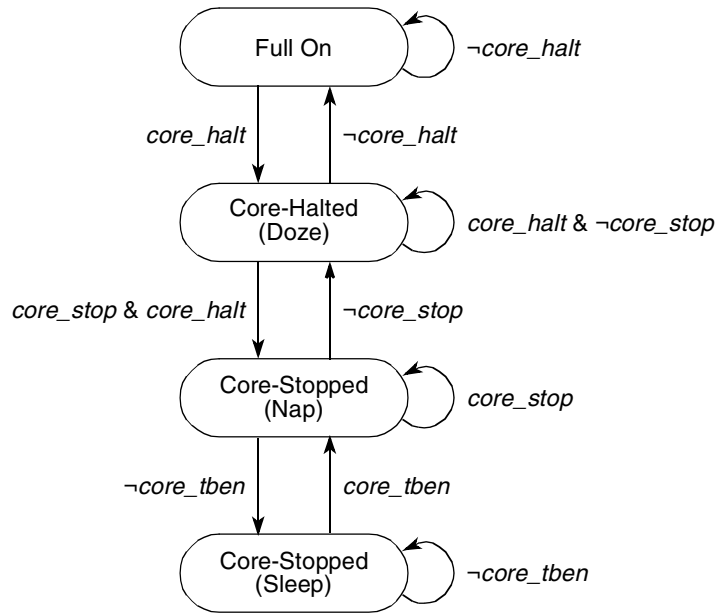


Figure 19-26. e500 Core Power Management State Diagram

For each operating state represented in the diagram, the core’s state is listed first, with the corresponding state of the MPC8533E shown beneath it in parenthesis. Note that there are many other variables that control the state transitions between MPC8533E power management states. These additional variables are described in more detail in Section 19.5.1.7, “Power-Down Sequence Coordination.”

Table 19-29 lists basic characteristics of the low-power modes and the full on mode.

Table 19-29. MPC8533E Power Management Modes—Basic Description

Mode	Description	Core Responds To		Signal States	
		Snoop	Interrupts	READY	ASLEEP
Full On	All units operating normally.	Yes	Yes	Asserted	Negated
Doze	Core stops dispatching new instructions (core is halted)	Yes	Yes	Negated	Negated

Table 19-29. MPC8533E Power Management Modes—Basic Description (continued)

Mode	Description	Core Responds To		Signal States	
		Snoop	Interrupts	READY	ASLEEP
Nap	Core is stopped with clocks off except to time base Should flush data cache before entering	No	Yes	Negated	Negated
Sleep	Core is stopped with clocks off. Clocks powered down to all blocks (including core time base) except to the interrupt controller (PIC) unit	No	Yes	Negated	Asserted

### 19.5.1.2 CKSTP\_IN is Not Power Management

CKSTP\_IN is not described here because it is not considered a power management signal, although asserting it does stop the core and a stopped core is technically in a low-power mode. CKSTP\_IN is described in [Section 19.3.2, “Detailed Signal Descriptions.”](#)

### 19.5.1.3 Dynamic Power Management

Many blocks in the MPC8533E can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

### 19.5.1.4 Shutting Down Unused Blocks

As described in [Section 19.4.1.12, “Device Disable Register \(DEVDISR\),”](#) DEVDISR provides a way to shut down certain functional blocks within the MPC8533E when they are not needed in a particular system. DEVDISR can be written by the e500 core or by an external master. Powering down a block in this way turns off all clocks to that block.

DEVDISR was designed with the expectation that, once initialized by software, it would be modified only by a hard system reset (HRESET). It is recommended that this register be written only during system initialization. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. (Setting DEVDISR[TB] disables the core’s timer facilities, and setting DEVDISR[E500] places the core in the core\_stopped state in which it does not respond to interrupts.) The results of re-enabling previously disabled blocks (by clearing the corresponding DEVDISR field) without a hard reset are boundedly undefined.

#### NOTE

Functional blocks disabled using DEVDISR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

### 19.5.1.5 Software-Controlled Power-Down States

e500 software can place the device in doze, nap, or sleep power-down states by writing to HID0 in the core. In addition, external masters can write to the memory-mapped POWMGTCR in the MPC8533E to cause the device to enter doze or sleep modes.

### 19.5.1.5.1 Doze Mode

In doze mode, the e500 core suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1 data cache is still supported and thus the data in the data cache is kept coherent. Interrupts directed to the core as described in [Section 10.1.3, “Interrupts to the Processor Core,”](#) are monitored by the device and cause the MPC8533E to use the defined handshake mechanism to exit the core from doze mode to allow the core to recognize and process the interrupt; however, unless the interrupt subroutine turns off (or masks) the control bits that enabled doze mode (MSR[WE], and HID0[DOZE]), the device re-enters doze mode after the interrupt has been serviced. See [Section 19.5.1.8, “Interrupts and Power Management,”](#) for more information.

The e500 core’s timer facilities are still enabled during doze mode, and core time base interrupts can be generated. All device logic external to the core remains fully operational in doze mode.

### 19.5.1.5.2 Nap Mode

In nap mode all clocks internal to the e500 core are turned off except for its timer facilities clock (the core time base). The L1 caches do not respond to snoops in nap mode, so if coherency with external I/O transactions is required, the L1 cache must be flushed before entering nap mode.

Similar to doze mode, interrupts occurring in nap mode cause the device to wake up the e500 core in order to service the interrupt. However, unless the interrupt service routine changes the control bits that caused the device to enter nap mode (MSR[WE], and HID0[NAP]), the MPC8533E returns to nap mode after the interrupt is serviced. See [Section 19.5.1.8, “Interrupts and Power Management,”](#) for more information.

All device logic external to the e500 core remains fully operational in nap mode.

### 19.5.1.5.3 Sleep Mode

In sleep mode, all clocks internal to the e500 core are turned off, including the timer facilities clock. All I/O interfaces in the device logic are also shut down. Only the clocks to the MPC8533E PIC are still running so that an external interrupt can wake up the device.

After the core and I/O interfaces have shut down, ASLEEP is asserted and READY is negated.

#### NOTE

Only external interrupts can wake the MPC8533E from sleep mode. Internal interrupt sources like the core interval timer or watchdog timer depend on an active clock for their operation and these are disabled in sleep mode.

## 19.5.1.6 Power Management Control Fields

The e500 core provides the following fields to signal power management requests to the MPC8533E device logic.

- MSR[WE]—Used to qualify the values of HID0[DOZE,NAP,SLEEP] in the generation of the internal *doze*, *nap*, and *sleep* signals.
- HID0[DOZE]—Signals the MPC8533E to initiate doze mode.



- HID0[NAP]—Signals the MPC8533E to initiate nap mode.
- HID0[SLEEP]—Signals the MPC8533E to initiate sleep mode.

These register fields and their functional relationship are shown in [Figure 19-27](#). The *PowerPC e500 Core Reference Manual* has details on accessing these power management control bits.

An external master can also initiate power management requests by setting the DOZ or SLP bits in the memory-mapped power management control and status register (POWMGTCSR). Because the core responds to snoops while dozing but not while napping, maintaining cache coherency requires significant preparation by the core before entering nap mode. For this reason only the core can initiate a nap during normal operation while other masters can initiate a doze.

### 19.5.1.7 Power-Down Sequence Coordination

To preserve cache coherency and otherwise avoid loss of system state, the core's transition to low-power modes is coordinated by a set of handshaking signals, shown in [Figure 19-27](#), and protocols with all other MPC8533E functional blocks that respond to power-down requests. The mode-transition protocol is executed automatically under these conditions and is shown in [Figure 19-26](#) and described in [Table 19-30](#).

The column in [Table 19-30](#) showing the global utilities block as initiating a low-power mode corresponds to the external masters that can write to the POWMGTCR that resides in the global utilities block. For the MPC8533E, these are the PCI interfaces. However, note that the core can also write to POWMGTCR and, in this case, can initiate power management through the global utilities block.

**Table 19-30. Power Management Entry Protocol and Initiating Functional Units**

Low-Power Mode	Entry Protocol	Initiating Functional Unit	
		Global Utilities	Core
Doze	<ol style="list-style-type: none"> <li>1. Assert <i>core_halt</i> input to core.</li> <li>2. Wait for <i>core_halted</i> handshake from core.</li> </ol>	√	√
Nap	<ol style="list-style-type: none"> <li>1. Follow doze protocol</li> <li>2. Assert <i>core_stop</i> input to core.</li> <li>3. Wait for <i>core_stopped</i> handshake from core.</li> </ol>	—	√
Sleep	<ol style="list-style-type: none"> <li>1. Follow doze protocol; send stop requests to rest of device.</li> <li>2. Follow nap protocol.</li> <li>3. Wait for all interfaces to acknowledge stop requests.</li> <li>4. Assert ASLEEP, negate READY, power down all clocks except to PIC unit.</li> </ol>	√	√

As shown in [Figure 19-27](#), the e500 core enters low-power modes only in response to the *core\_halt*, *core\_stop*, or *core\_tben* inputs from the MPC8533E's power management logic. These inputs may be prompted by the core (by setting the NAP, DOZE, or SLEEP bits in the HID0 when enabled by setting MSR[WE]) or by an external master (by setting POWMGTCR[DOZ,SLP]).

[Figure 19-27](#) shows how all the clocking to the core timer facilities is disabled by clearing HID0[TBEN]. When enabled, (HID0[TBEN] = 1), the clock source is either the CCB clock divided by eight (the default) or a synchronized version of the RTC input. For more details, see [Section 6.10.1, “Hardware Implementation-Dependent Register 0 \(HID0\).”](#)

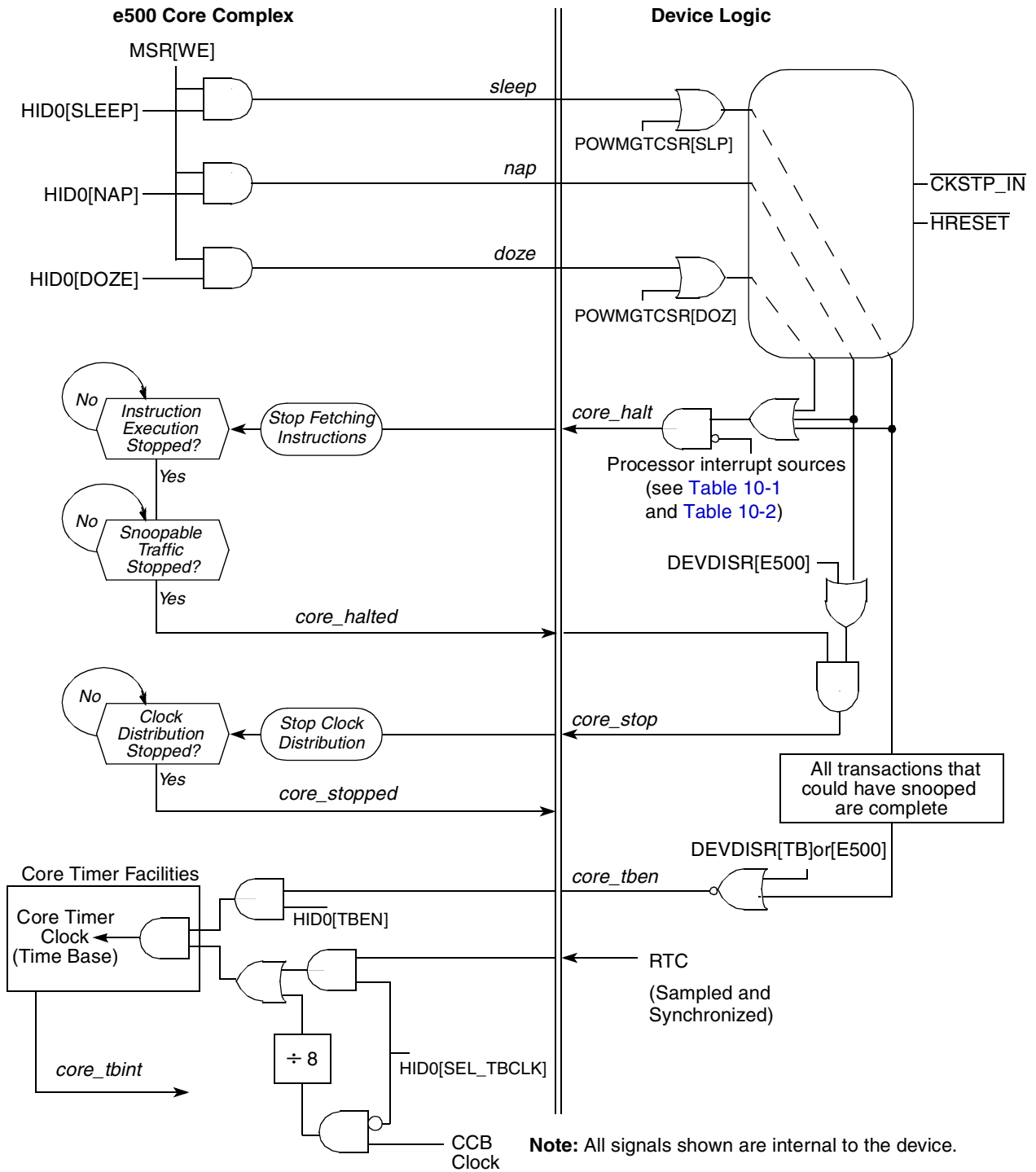


Figure 19-27. MPC8533E Power Management Handshaking Signals

## 19.5.1.8 Interrupts and Power Management

Whether low-power modes are automatically re-enabled after an interrupt is processed differs depending on whether the low power mode was entered due to a write to the core MSR[WE] bit or the low power mode was entered due to a write to POWMGTCR.

### 19.5.1.8.1 Interrupts and Power Management Controlled by MSR[WE]

When an interrupt is asserted to the CPU, the core complex saves portions of the MSR to MCSRR1, CSRR1, or SRR1 (depending on the type of interrupt), and restores those values on return from the routine. MSR[WE], which gates the *doze*, *nap*, and *sleep* power management outputs (internal device signals) from the core complex, is always among the bits saved and restored; hence these outputs negate to the MPC8533E power management logic when the interrupt begins processing in the core. They return to their previous state when the core executes an **rfi**, **rfdi**, or **rfmci** instruction. [Section 10.1.3, “Interrupts to the Processor Core,”](#) lists interrupts that cause the MPC8533E to wake up.

#### NOTE

Returning *doze*, *nap*, and *sleep* signals to their original state when MSR[WE] is restored differs from how power management is implemented on earlier devices based on PowerPC ISA where MSR[POW], which enables power-down requests, is cleared when the processor exits a low-power state and is not automatically restored, as it is in Power ISA implementations.

### 19.5.1.8.2 Interrupts and Power Management Controlled by POWMGTCR

The IRQ\_MSK and CI\_MSK fields of the POWMGTCR register prevent  $\overline{int}$  interrupts or  $\overline{cint}$  critical interrupts from waking the device from a low power state. This is true regardless of the method used to enter the low power state.

Any unmasked interrupt (not masked by the mask bits in the POWMGTCR register) causes the POWMGTCR[DOZ,SLP] fields to be cleared when it occurs. When such an interrupt occurs, the device returns to the normal operating mode and does not automatically attempt to return to a low power state after the interrupt is handled.

Note that interrupts caused by the unconditional debug event ( $\overline{UDE}$ ) and machine check ( $\overline{MCP}$ ) signals are not masked by the IRQ\_MSK and CI\_MSK fields; therefore, when these signals assert, the POWMGTCR[DOZ,SLP] fields are cleared and the device will return to full power operation. See [Section 19.4.1.13, “Power Management Control and Status Register \(POWMGTCR\),”](#) for detailed information about the bits of POWMGTCR.

Note also that unmasked interrupts that occur while the device is in the process of going into the sleep state (before sleep is completely attained) can also cause the device to clear the POWMGTCR[DOZ,SLP] fields and return the device to full power operation.

### 19.5.1.9 Snooping in Power-Down Modes

When the MPC8533E is in doze mode, the e500 core is in the core-halted state and it snoops its L1 caches and full coherency is maintained. In deeper power-down modes, however, the e500 core does not respond to snoops.

The MPC8533E does not perform dynamic bus snooping as described in the *e500 Reference Manual*. That is, when the e500 core is in the core-stopped state (which is the state of the core when the MPC8533E is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, the L1 caches should be flushed if coherency is required during these power-down modes.

### 19.5.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8533E logic (external to the core complex) to enter a power saving state. It is assumed that the desired power-saving state (doze, nap, or sleep) was set up by setting the appropriate HID0 bit, typically at system start-up time. Setting WE has no direct effect on instruction execution, but is reflected on the internal *doze*, *nap*, and *sleep* signals, depending on the HID0 settings. To ensure a clean transition into and out of a power-saving mode, the following program sequence is recommended:

```

        sync
        mtmsr (WE)
        isync
loop:   br loop

```

### 19.5.1.11 Requirements for Reaching and Recovering from Sleep State

In order to successfully reach the sleep state, I/O traffic to the device must be stopped. The logic that controls the power down sequence waits for all I/O interfaces to become idle. In some applications this may happen eventually without actively shutting down interfaces, but most likely, software will have to take steps to shut down the eTSEC, and PCI interfaces before issuing the command (either the write to the core MSR[WE] as described above or writing to POWMGTCR) to put the device into sleep state.

The PCI interfaces will begin retrying inbound transactions before entering a power down state. The PCI interfaces, however, could potentially be in an unknown state when they exit sleep if they were in the middle of a retry sequence when internal clocks were shut down. Therefore it is strongly recommended that system software clear the memory space bit in the PCI Bus Command Register before putting the device in sleep mode. Software may also need to set the Agent Config Lock bit of the PCI Bus Function Register so that the device will not respond to configuration transactions. Upon exiting sleep mode, software should return these configuration bits to their normal state.

## 19.5.2 General-Purpose I/O Signals

When configured as general-purpose I/O signals, software can read inputs by reading the associated GPIO data register (See [Section 19.4.1.10, “General-Purpose Input Data Register \(GPINDR\).”](#)). Output values can be set by writing to the associated GPIO data register (See [Section 19.4.1.9, “General-Purpose Output Data Register \(GPOUTDR\).”](#)). For details regarding the control and status of the general-purpose I/O signals, see [Section 19.4.1.8, “General-Purpose I/O Control Register \(GPIOCR\).”](#)

As well, a general purpose input register is loaded with the values of the local bus address/data pins at the negation of  $\overline{\text{HRESET}}$ . See [Section 19.4.1.7, “General-Purpose POR Configuration Register \(GPPORCR\),”](#) for additional details.

### NOTE

Unused  $\text{IRQ}_n$  signals may also be used as general-purpose inputs. The external interrupt summary register (ERQSR) can be used to monitor these signals. See [Section 10.3.3.1, “External Interrupt Summary Register \(ERQSR\),”](#) for more information.

## 19.5.3 Interrupt and Local Bus Signal Multiplexing

The MPC8533E has very little signal multiplexing. Two sets of DMA channel triggering signals can alternately be placed on other signals as follows:

- $\overline{\text{LCS}}[5:7]$  are multiplexed with DMA channel 2  $\overline{\text{DMA\_DREQ2}}$ ,  $\overline{\text{DMA\_DACK2}}$ , and  $\overline{\text{DMA\_DONE2}}$ .
- $\text{IRQ}[9:11]$  are multiplexed with DMA channel 3  $\overline{\text{DMA\_DREQ3}}$ ,  $\overline{\text{DMA\_DACK3}}$ , and  $\overline{\text{DMA\_DDONE3}}$ .

For details regarding the selection of the alternate function DMA trigger, see [Section 19.4.1.11, “Alternate Function Signal Multiplex Control Register \(PMUXCR\).”](#)



## Chapter 20

# Device Performance Monitor

This chapter describes the device performance monitor facility, which can be used to monitor and optimize performance. The e500 core implements a separate performance monitor for strictly core-related behavior, such as instruction timing and L1 cache operations. This is described in the *PowerPC e500 Core Reference Manual* (Freescale Document Order No. E500CORERM).

[Section 20.4.7, “Performance Monitor Events,”](#) briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

### 20.1 Introduction

The device-level performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. Although the performance monitor described here is similar in many respects to the performance monitor facility implemented on the e500 core, it differs in that it is implemented using memory-mapped registers and it counts events outside the e500 core, for example, PCI, DDR, and L2 cache events.

Performance monitor counters (PMC0–PMC9) are used to count events selected by the performance monitor local control registers. PMC0 is a 64-bit counter specifically designated to count cycles. PMC1–PMC9 are 32-bit counters that can monitor 64 counter-specific events in addition to counting 64 reference events.

The benefits of the on-chip performance monitor are numerous, and include the following:

- Because some systems or software environments are not easily characterized by signal traces or benchmarks, the performance monitor can be used to understand the device’s behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

### 20.1.1 Overview

Figure 20-1 is a high-level block diagram of the performance monitor, which consists of a global control register (PMGC0), one 64-bit counter (PMC0), nine 32-bit counters, and two control registers per counter (20 total control registers). The global control register PMGC0 affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values, and the value of the threshold multiplier. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

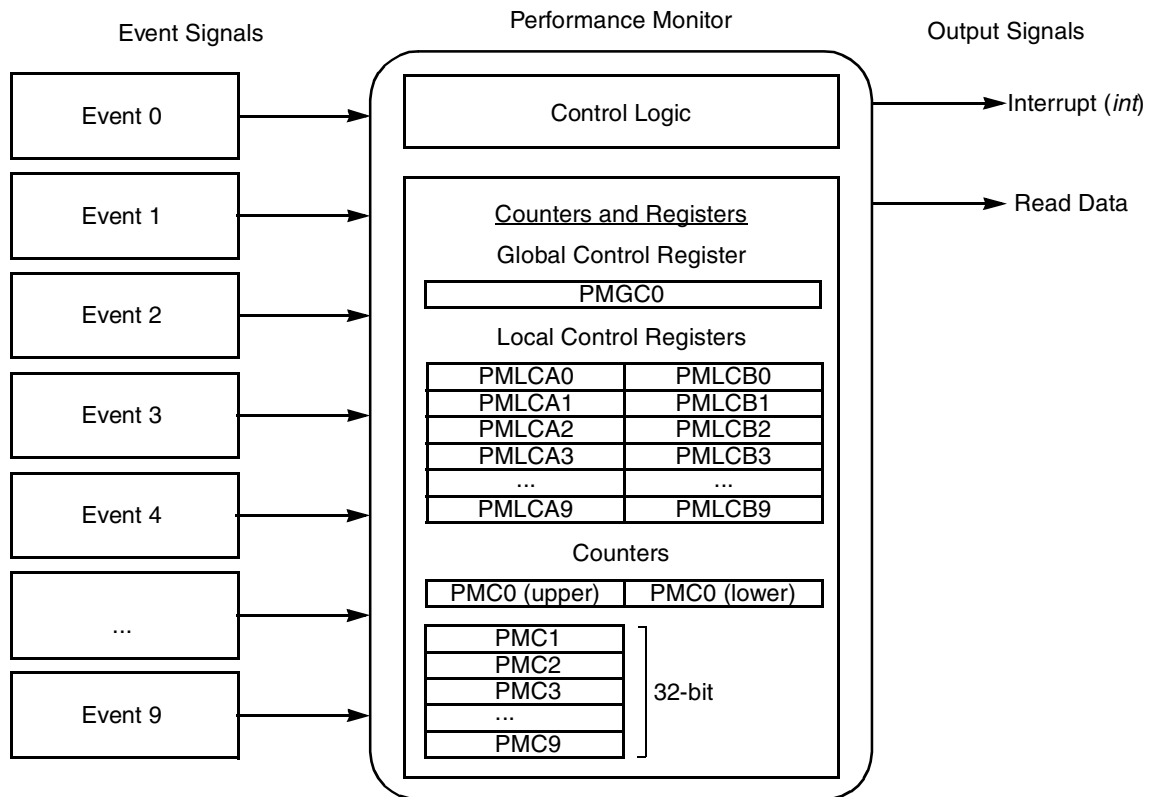


Figure 20-1. Performance Monitor Block Diagram

Performance monitor events are signalled by the functional blocks in the integrated device and are selectively recorded in the PMCs. Sixty-four of these events are referred to as reference events, which can be counted on any of the nine 32-bit counters. Counter-specific events can be counted only on the counter where the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signalled. The PMCs can also be programmed to freeze when an interrupt is signalled.



## 20.1.2 Features

The performance monitor offers a rich set of features that permits a complete performance characterization of the implementation. These features include:

- One 64-bit counter exclusively dedicated to counting cycles
- Nine 32-bit counters that count the occurrence of selected events
- One global control register (affects all counters) and two local control registers per counter
- Ability to count up to 64 reference events that may be counted on any of the nine 32-bit counters
- Ability to count up to 576 counter-specific events
- Triggering and chaining capability
- Duration and quantity threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

## 20.2 Signal Descriptions

The performance monitor does not have any signals that are driven externally (off-chip) but it does assert the internal interrupt (*int*) signal on a performance monitor interrupt condition.

## 20.3 Memory Map and Register Definition

Performance monitor registers reside in the run-time register block starting at offset 0xE\_1000. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved. This section describes the registers implemented to support the performance monitor facilities. [Table 20-1](#) lists the performance monitor registers. These registers can be read or written only with 32-bit accesses.

### 20.3.1 Register Summary

The performance monitor uses ten counter registers and a group of local control registers that are used to specify the method of counting. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 20-1. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	<a href="#">20.3.2.1/20-5</a>
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1018	PMC0 (lower)—Performance monitor counter 0 upper	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_101C	PMC0 (upper)—Performance monitor counter 0 lower	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_10A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>

**Table 20-1. Control Register Memory Map (continued)**

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_10A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_10A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>

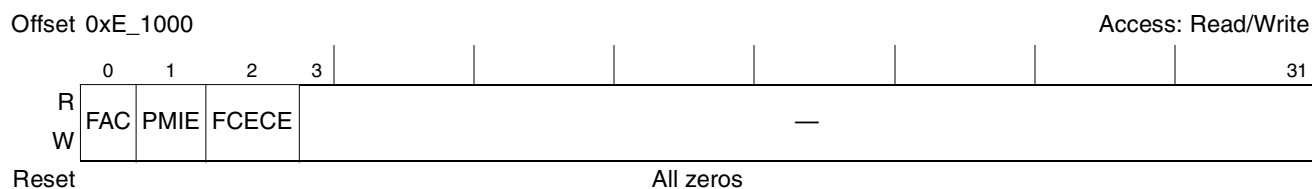
In addition to these registers, the interrupt control provides four pairs of mask registers that can be used to monitor message, interprocessor, timer, and external interrupts. See [Section 10.3.4, “Performance Monitor Mask Registers \(PMMRs\),” on page 10-32](#).

## 20.3.2 Control Registers

This section describes the performance monitor control registers in detail.

### 20.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in [Figure 20-2](#), is a 32-bit register used to control all PMCs.

**Figure 20-2. Performance Monitor Global Control Register (PMGC0)**

[Table 20-2](#) describes PMGC0 fields.

**Table 20-2. PMGC0 Field Descriptions**

Bits	Name	Description
0	FAC	Freeze all counters. 0 PMCs are incremented (if permitted by other PMGC0/PMLC bits). 1 PMCs are not incremented. Set by hardware when an interrupt is signalled and FCECE =1.
1	PMIE	Performance monitor interrupt enable. Interrupts are caused by PMC overflows. 0 Interrupts are disabled. 1 Interrupts are enabled and occur when an enabled condition or event occurs.
2	FCECE (DISCOUNT)	Freeze counters on enabled condition or event. An enabled condition or event is defined as: The msb = 1 in $PMC_n$ and $PMLCA_n[CE] = 1$ . The use of the trigger and freeze counter conditions depends on the enabled condition. 0 PMCs can be incremented (if permitted by other control bits). 1 PMCs can be incremented (if permitted by other control bits) only until an enabled condition or event occurs, at which time PMGC0[FAC] is set. It is up to software to clear FAC.
3–31	—	Reserved

### 20.3.2.2 Performance Monitor Local Control Registers (PMLCA<sub>n</sub>, PMLCB<sub>n</sub>)

The performance monitor local control registers (PMLCA<sub>n</sub> and PMLCB<sub>n</sub>) are used to control the operation of the PMCs. The performance monitor local control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter.

Figure 20-3 shows the performance monitor local control A0 register (PMLCA0).

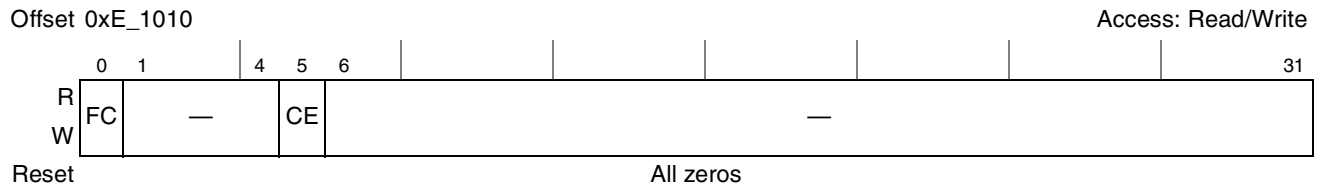


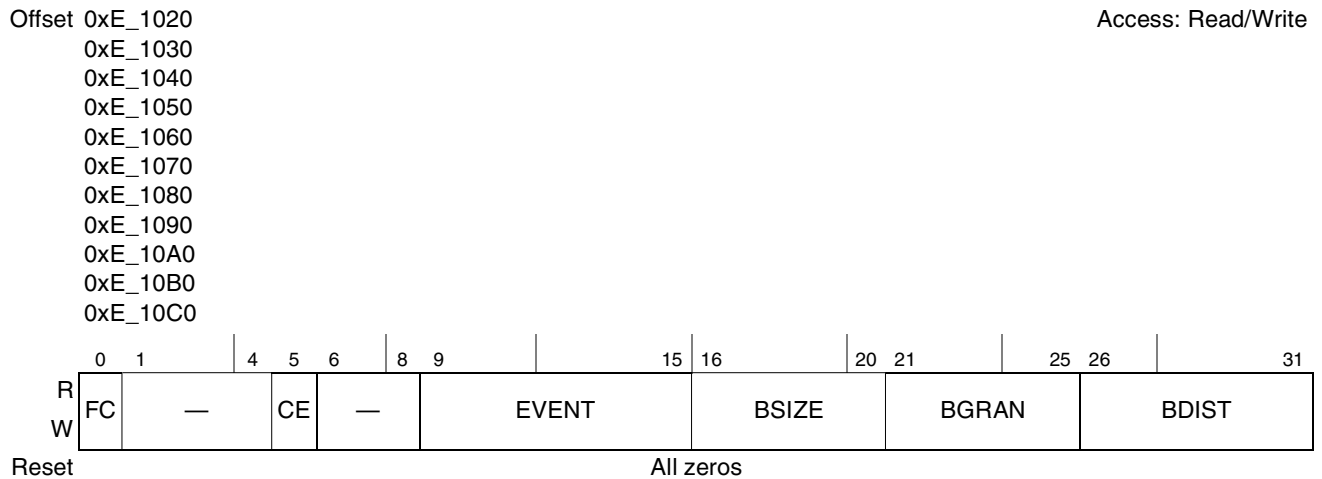
Figure 20-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 20-3 describes PMLCA0 fields.

Table 20-3. PMLCA0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter. Basic counter enable. 0 The PMCs are enabled and incremented (if permitted by other SPM control bits). 1 The PMCs are disabled—they do not increment.
1–4	—	Reserved
5	CE	Condition enable. Controls counter overflow condition. Should be cleared when PMC0 is used as a trigger or is selected for chaining. 0 Overflow conditions for PMC0 cannot occur (PMC0 cannot cause interrupts or freeze counters) 1 Overflow conditions occur when PMC0[msb] is set.
6–31	—	Reserved

Figure 20-4 shows the performance monitor local control registers A1–A9.



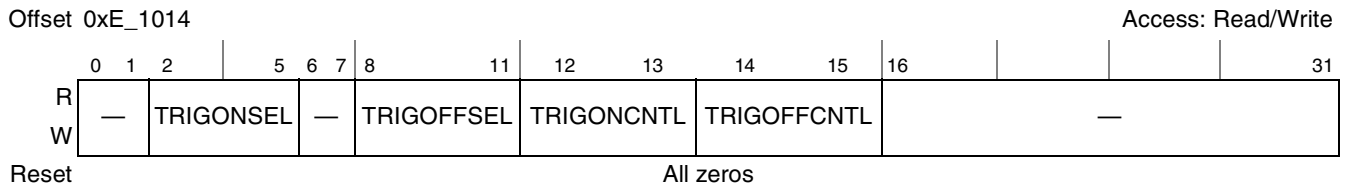
**Figure 20-4. Performance Monitor Local Control A Registers (PMLCA1–PMLCA9)**

Table 20-4 describes PMLCA $n$  fields.

**Table 20-4. PMLCA1–PMLCA9 Field Descriptions**

Bits	Name	Description
0	FC	Freeze counter 0 The PMCs are incremented (if permitted by other PMC control bits). 1 The PMCs are not incremented (if permitted by other PMC control bits).
1–4	—	Reserved
5	CE	Condition enable 0 Overflow conditions for PMC $n$ cannot occur (PMC $n$ cannot cause interrupts or freeze counters). Should be cleared when PMC $n$ is used as a trigger or is selected for chaining. 1 Overflow conditions occur when PMC $n$ [msb] is set.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable. Note that with counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64. See Table 20-10 for definition of events.
16–20	BSIZE	Burst size. Fewest event occurrences that constitute a burst, that is, a rapid sequence of events followed by a relatively long pause. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion. See Section 20.4.6, “Burstiness Counting,” for more information.
21–25	BGRAN	Burst granularity. The maximum number of clock cycles between events that are considered part of a single burst. See Section 20.4.6, “Burstiness Counting.”
26–31	BDIST	Burst distance (used with TBMULT). The number of clock cycles between bursts. Must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 20-5 shows the performance monitor local control B0 register (PMLCB0).



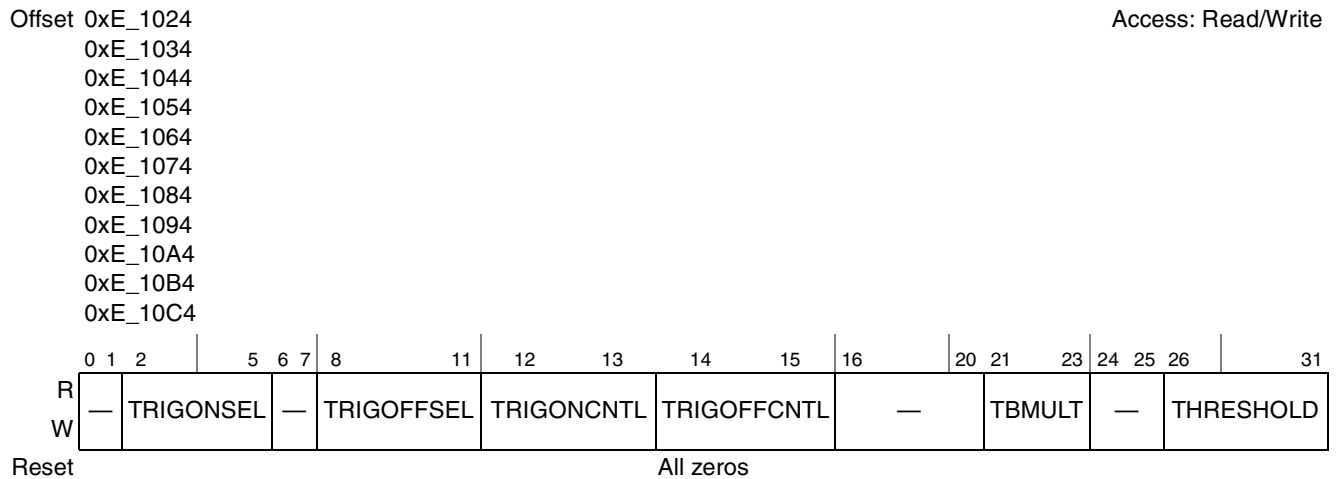
**Figure 20-5. Performance Monitor Local Control Register B0 (PMLCB0)**

Table 20-5 describes PMLCB0 fields.

**Table 20-5. PMLCB0 Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. The number of the counter that starts event counting. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. The number of the counter that stops event counting. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–31	—	Reserved

Figure 20-6 shows performance monitor local control registers 1–9.



**Figure 20-6. Performance Monitor Local Control Register B (PMLCB1–PMLCB9)**

Table 20-6 describes PMLCB $n$  fields.

**Table 20-6. PMLCB $n$  Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. Set this field equal to the number of the counter that should trigger event counting to start. When the specified counter’s TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. Set this field equal to the number of the counter that should trigger event counting to stop. When the specified counter’s TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–20	—	Reserved

**Table 20-6. PMLCB<sub>n</sub> Field Descriptions (continued)**

Bits	Name	Description
21–23	TBMULT	Threshold and burstiness multiplier. Threshold events are counted when the event duration exceeds a specified threshold value. The threshold is scaled based on the TBMULT settings. TBMULT is not used to scale the threshold value for quantity threshold events. The burst distance for burstiness counting is also scaled using the TBMULT settings. For all events that scale the threshold, the threshold field is multiplied by the factors shown below (ranging from 1 to 128). 000 1 001 2 010 4 011 8 100 16 101 32 110 64 111 128
24–25	—	Reserved
26–31	THRESHOLD	Threshold. Only events whose (number of) occurrences exceed this value are counted. By varying the threshold value, software can characterize the events subject to the threshold. For example, if PMC2 counts eTSEC BD read latencies for which the duration exceeds the threshold, software can obtain the distribution of eTSEC BD read latencies for a given program by monitoring the program using various threshold values.

### 20.3.3 Counter Registers

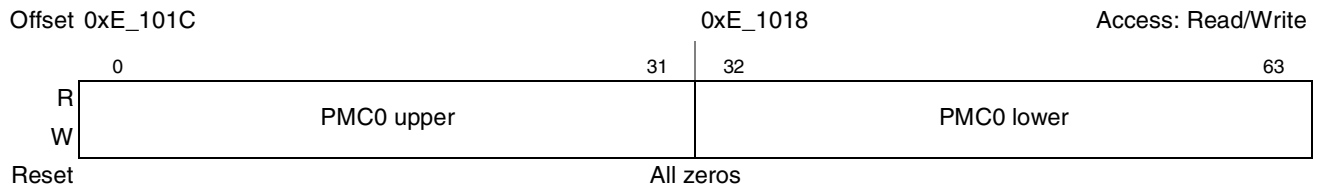
This section describes the PMCs in detail.

#### NOTE

Because accessing a PMC manually has priority over incrementing it due to event counting, writing a PMC while it is counting may affect the count. Likewise, writing a performance monitor control register while its target counter is counting may also affect the count.

#### 20.3.3.1 Performance Monitor Counters (PMC0–PMC9)

PMC0–PMC9 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 20-7, is associated with two 32-bit registers that form a 64-bit counter designated to count clock cycles. PMC0 upper represents the upper 32 bits of counter 0, and PMC0 lower represents the lower 32 bits.



**Figure 20-7. Performance Monitor Counter Register 0 (PMC0)**

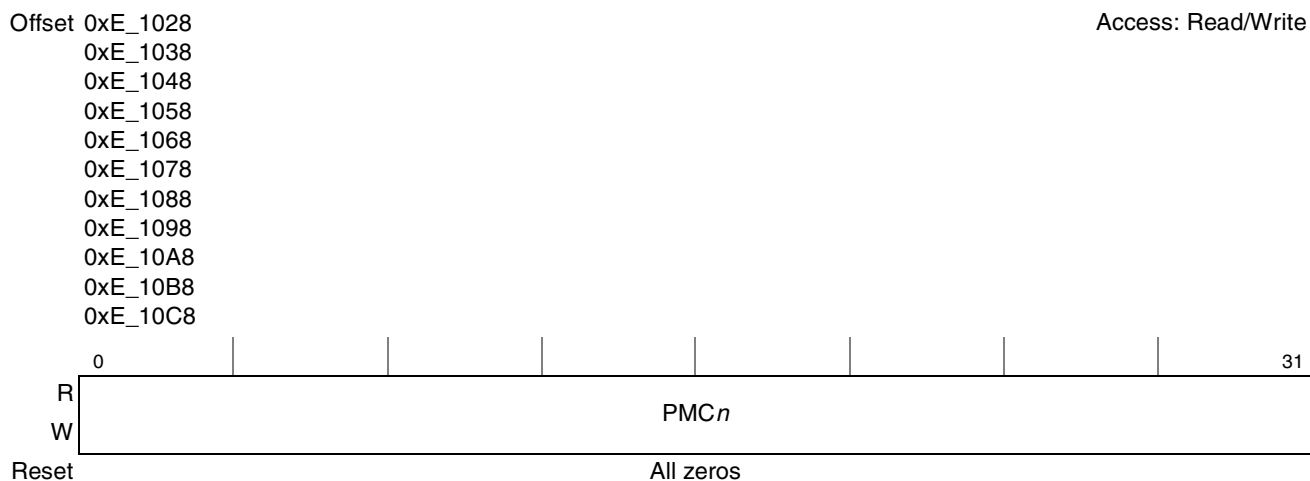


Table 20-7 describes PMC0 fields.

**Table 20-7. PMC0 Field Descriptions**

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC9, shown in Figure 20-8, are 32-bit counters that can monitor 64 unique events in addition to the 64 reference events that can be counted on all of these registers.



**Figure 20-8. Performance Monitor Counter Register (PMC1–PMC9)**

Table 20-8 describes PMC<sub>n</sub> fields.

**Table 20-8. PMC[1–9] Field Descriptions**

Bits	Name	Description
0–31	PMC <sub>n</sub>	Event count. An overflow is indicated when the msb = 1. Manually setting the msb can cause an immediate interrupt.

## 20.4 Functional Description

This section describes the use of some features of the performance monitor.

### 20.4.1 Performance Monitor Interrupt

PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit (PMLCAN[CE]) and performance monitor interrupt enable bit (PMGC0[PMIE]) must be set. When an interrupt is signalled and the freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set, PMGC0[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

## 20.4.2 Event Counting

Using the control registers described in [Section 20.3.2, “Control Registers,”](#) the twelve PMCs can count the occurrences of specific events. The 64-bit PMC0 is designated to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC9). Additionally, up to 64 unique events can be counted on each 32-bit counter.

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAN[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.

Note that using PMLCAN[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

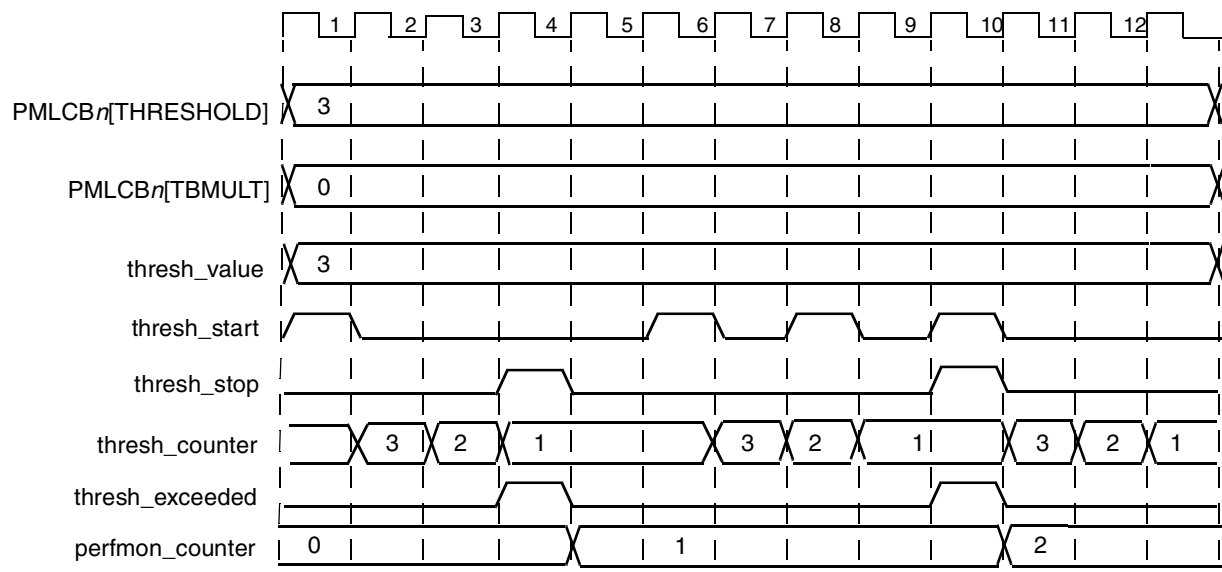
## 20.4.3 Threshold Events

The threshold feature allows characterization of events that can take a variable number of clock cycles to occur. Threshold events are counted only if the latency is greater than the threshold value specified in PMLCB $n$ [THRESHOLD]. There are two types of threshold events.

The first type of threshold events are duration threshold events. For duration threshold event sequences, the PMC increments only when the duration of the event is equal to or greater than the threshold value. The threshold value is scaled by a multiple specified in PMLCB $n$ [TBMULT].

A duration threshold event requires two signals: The first indicates when a threshold event sequence begins, and the second indicates when it ends. An internal counter determines when the threshold count is exceeded and when the PMC can increment. This internal counter decrements during a threshold event sequence until it reaches the value of one. A new sequence cannot begin until the current one completes. Additional threshold start signals are ignored during a sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle in a current sequence, the stop terminates the current sequence and the start signals the beginning of a new one. However, if both signals are asserted during the same cycle while not in a current event sequence, both signals are ignored. [Figure 20-9](#) is a timing diagram for duration threshold event counting.

An illegal condition exists if the threshold value obtained from PMLCB $n$ [THRESHOLD] and PMLCB $n$ [TBMULT] is less than two. Under these conditions the intent of threshold counting is ambiguous.



<sup>1</sup> For this example a threshold value of three indicates that the user wishes to count the number of times a particular event lasts three cycles or longer.

**Figure 20-9. Duration Threshold Event Sequence Timing Diagram**

The second type of threshold event is the quantity threshold event. For these types of threshold event sequences the performance monitor counter is only incremented when the specified threshold event exceeds the threshold value. These events do not use the multiplier register field (PMLCBn[TBMULT]) like the duration threshold events. This type of threshold event is generally used to monitor the usage of buffers and queues. For example, the usage of a specific queue could be characterized by measuring the amount of time the queue is completely full or partially full. For this example the threshold field would be used to specify how many entries are required to be valid in the queue for that event to be counted.

## 20.4.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in [Table 20-10](#), are dedicated to specific registers.

Note that the chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register's msb is set. For this overflow to occur, PMLCAn[CE] should be cleared to avoid signalling an interrupt when the counter's most significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

## 20.4.5 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local control register of counter

PMC1. This is done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs.

When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions.

Triggering is disabled when the counter’s trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

### 20.4.6 Burstiness Counting

The burstiness counting feature makes it easier to characterize events that occur in rapid succession followed by a relatively long pause. As shown in Table 20-9, event bursts are defined by size, granularity, and distance.

Table 20-9. Burst Definition

Parameter	Description	Register Field
Size	The minimum number of events constituting a burst	PMLCA <sub>n</sub> [BSIZE]
Granularity	The maximum time between individual events counted as members of the same burst	PMLCA <sub>n</sub> [BGRAN]
Distance	The minimum time between bursts	PMLCA <sub>n</sub> [BDIST] x PMLCB <sub>n</sub> [TBMULT]

Figure 20-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

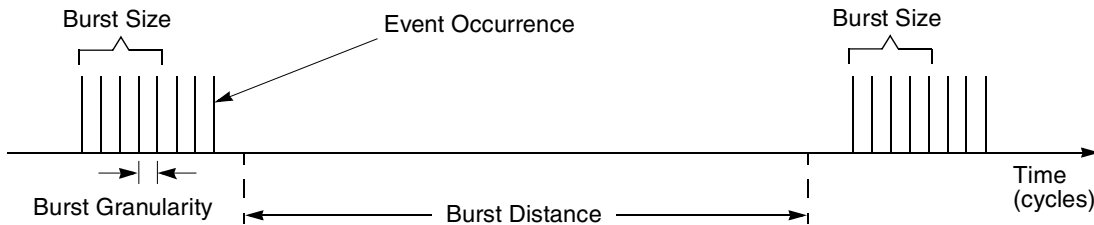


Figure 20-10. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA<sub>n</sub>[BSIZE]) specifies the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences equals or exceeds PMLCA<sub>n</sub>[BSIZE]. Furthermore, these individual event occurrences must be separated by no more clock cycles than the value in the burstiness granularity field (PMLCA<sub>n</sub>[BGRAN]). Note that, although a burst is identified when the minimum number of events occurs, it is not counted until the burst sequence has ended. A burst sequence ends when the specified burstiness granularity is exceeded, at which point the last valid event has occurred for that sequence.

PMLCAn[BGRAN] specifies the maximum number of cycles between individual events for them to qualify as members of the same burst sequence.

The burstiness distance field (PMLCAn[BDIST]) and threshold/burstiness multiplier field (PMLCBn[TBMULT]) specify the acceptable number of cycles between the end of a burst sequence and the beginning of a new sequence for a group of event occurrences to be counted as an individual burst. The product of the burstiness distance field and the threshold/burstiness multiplier field determine the burstiness distance value used to determine when another burst sequence can begin. Note that the burst distance count begins when a new burst sequence ends and the PMC is incremented. No new burst sequence may begin until the burst distance count has reached zero. After the burst distance count reaches zero, it holds the zero value indicating that a new burst sequence can be counted. The burst distance count begins again when a new burst sequence is identified and counted.

Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

[Figure 20-10](#) shows that the burst distance is measured from the end of one burst sequence and that a new burst sequence may not begin until the burst distance count expires.

Three internal counters track the different values required for burstiness counting.

- Burstiness size is monitored by a counter. It is loaded with the value specified in the local control register when the burst granularity counter and the burst distance counters reach zero, and no new event is occurring. It always decrements when the following conditions occur: its value is not already zero, an event occurs, and the burst distance count equals zero.
- Burstiness granularity is monitored by a counter that is loaded with the specified value in the local control register on the rising edge of an event occurrence whenever the burst distance count equals zero. The granularity counter is decremented (if it has not already reached zero) when an event is not occurring and burst distance count equals zero.
- Burstiness distance is measured by a counter that is loaded with the product of PMLCBn[BDIST] and PMLCBn[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled (and the counter has not already reached zero).

A burst is counted at the end of a burst sequence when the three burst parameter counters are all equal to zero. Figure 20-11 shows a burstiness counting example.

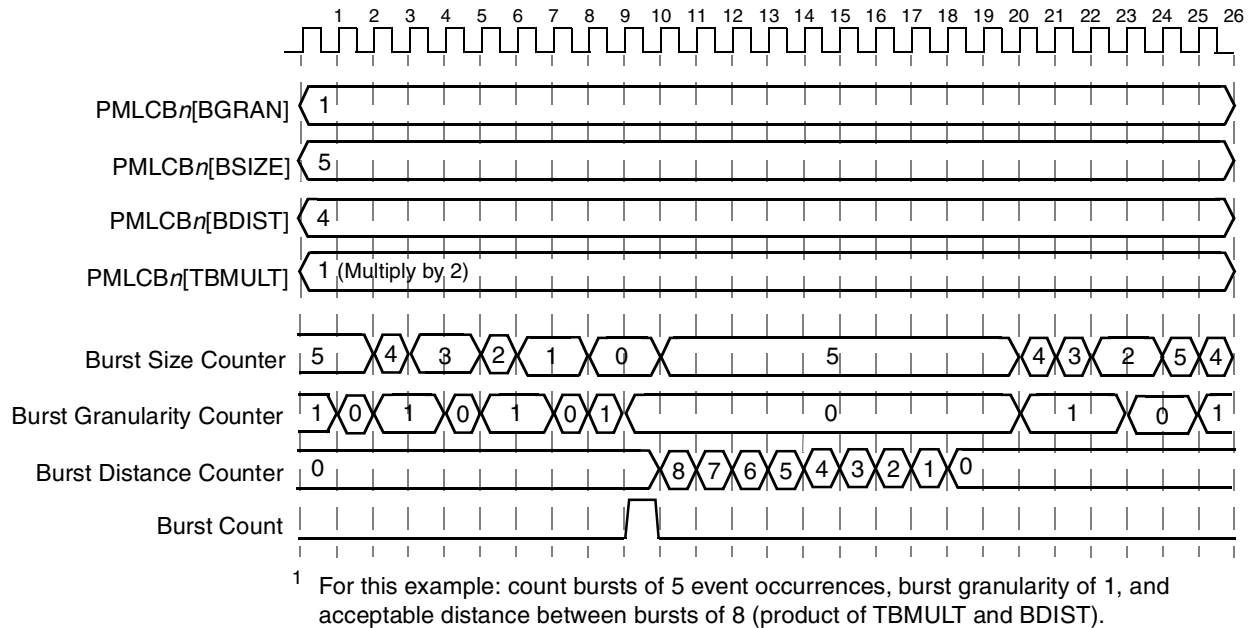


Figure 20-11. Burstiness Counting Timing Diagram

### 20.4.7 Performance Monitor Events

Table 20-10 lists performance monitor events specified in PMLCA1–PMLC9.

The event assignment column indicates the event’s type and number, using the following formats:

- Ref:#—Reference events are shared across counters PMC1–PMC9. The number indicates the event. For example, Ref:6 means that PMC1–PMC9 share reference event 6.
- C[0–9]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:126 means PMC8 is assigned event 126 (PIC interrupt wait cycles).

Counter events not specified in Table 20-10 are reserved.

#### NOTE

Events generated by peripherals attached to the on-chip network (DMA, PCI, PCI Express) are counted twice, because these peripherals run at a slower clock rate than peripherals connected directly to the peripheral bus (for example, DDR or eTSEC).

Table 20-10. Performance Monitor Events

Event Counted	Number	Description of Event Counted
<b>General Events</b>		
Nothing	Ref:0	Register counter holds current value
System cycles	C:64 and Ref:20	CCB (platform) clock cycles
<b>DDR Memory Controller Events</b>		
Cycles a read is returning data from DRAM	Ref:19	Each data beat returned to the memory controller on the DRAM interface
Cycles a read or write transfers data from (or to) DRAM	Ref:11	Each data beat transferred to or from the DRAM
Pipelined read misses in the row open table	C1:121	Row open table read misses issued while a read is outstanding
Pipelined read or write misses in the row open table	C2:64	Row open table read or write misses issued while a read or write is outstanding
Non-pipelined read misses in the row open table	C3:124	Row open table read misses issued when no reads are outstanding
Non-pipelined read or write misses in the row open table	C4:64	Row open table read or write misses issued when no reads or writes are outstanding
Pipelined read hits in the row open table	C5:120	Row open table read hits issued when a read is outstanding
Pipelined read or write hits in the row open table	C6:64	Row open table read or write hits issued when a read or write is outstanding
Non-pipelined read hits in the row open table	C7:121	Row open table read hits issued when no reads are outstanding
Non-pipelined read or write hits in the row open table	C8:64	Row open table read or write hits issued when no reads or writes are outstanding
Forced page closings not caused by a refresh	C1:64	Precharges issued to the DRAM for any reason except refresh. The possibilities are as follows: <ul style="list-style-type: none"> <li>• A new transaction must be issued to an already active bank and sub-bank that has a different row open.</li> <li>• A new transaction must be issued, but the row open table is full and there is no bank/sub-bank match between the current transaction and the row open table.</li> <li>• The BSTOPRE interval expired for an open row.</li> </ul>
Row open table misses	C2:65	Transactions that miss in the row open table
Row open table hits	C3:64	Transaction that hit in the row open table
Force page closings	C4:65	Forced page closings including those due to refreshes
Read-modify-write transactions due to ECC	C5:64	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence is issued on the DRAM interface.
Forced page closings due to collision with bank and sub-bank	Ref:12	Increments if a new transaction must be issued to an active bank and sub-bank that has a different row open
Reads or writes from core	Ref:13	—

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Reads or writes from eTSEC 1 or 3	C3:65	—
Reads or writes from high speed interfaces (PCI and PEX2)	C3:66	—
Reads or writes from high speed interfaces (PCI and PCI Express)	C4:67	—
Reads or writes from DMA	C5:66	—
Reads or writes from Security	C6:69	—
Row open table hits for reads or writes from core	Ref:14	—
Row open table hits for reads or writes from eTSEC 1 or 3	C6:65	—
Row open table hits for reads or writes from high speed interfaces (PCI and PEX2)	C6:66	—
Row open table hits for reads or writes from high speed interfaces (PCI and PCI Express)	C7:65	—
Row open table hits for reads or writes from DMA	C8:66	—
Row open table hits for reads or writes from Security	C7:68	—
<b>Memory Target Queue Events</b>		
MEM TQ read/write address collision	C5:69	—
<b>DMA Controller Events</b>		
Channel 0 read request	C1:66	DMA channel 0 read request active in the system
Channel 1 read request	C2:69	DMA channel 1 read request active in the system
Channel 2 read request	C3:68	DMA channel 2 read request active in the system
Channel 3 read request	C4:70	DMA channel 3 read request active in the system
Channel 0 write request	C1:67	DMA channel 0 write request active in the system
Channel 1 write request	C2:70	DMA channel 1 write request active in the system
Channel 2 write request	C3:69	DMA channel 2 write request active in the system
Channel 3 write request	C4:71	DMA channel 3 write request active in the system
Channel 0 descriptor request	C5:105	DMA channel 0 descriptor request active in the system
Channel 1 descriptor request	C6:108	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C7:105	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C8:105	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:68 and C5:117	DMA channel 0 read double word valid



Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Channel 1 read DW or less	C2:71 and C6:122	DMA channel 1 read double word valid
Channel 2 read DW or less	C3:70 and C7:118	DMA channel 2 read double word valid
Channel 3 read DW or less	C4:72 and C8:116	DMA channel 3 read double word valid
Channel 0 write DW or less	C1:69	DMA channel 0 write double word valid
Channel 1 write DW or less	C2:72	DMA channel 1 write double word valid
Channel 2 write DW or less	C3:71	DMA channel 2 write double word valid
Channel 3 write DW or less	C4:73	DMA channel 3 write double word valid
<b>e500 Coherency Module (ECM) Events</b>		
ECM request wait core	C8:77	Asserted for every cycle core request occurs
ECM request wait I <sup>2</sup> C/Security	C7:77	Asserted for every cycle I2C/Security request occurs
ECM request wait PCI//DMA	C5:80	Asserted for every cycle PCI/PEX/DMA request occurs
ECM request wait eTSEC1	C6:80	Asserted for every cycle eTSEC1 request occurs
ECM request wait eTSEC3	C4:84	Asserted for every cycle eTSEC3 request occurs
ECM dispatch	Ref:15	ECM dispatch (includes address only's) <b>Note:</b> All ECM dispatch events are for committed dispatches
ECM dispatch from core	C1:80	ECM dispatch from core (includes address only's)
ECM dispatch from eTSEC1	C3:83	—
ECM dispatch from eTSEC3	C8:80	—
ECM dispatch from PCI	C6:81	—
ECM dispatch from PEX2	C7:80	—
ECM dispatch from PEX1	C8:81	—
ECM dispatch from PEX3	C7:94	—
ECM dispatch from DMA	C7:78	—
ECM dispatch from Security	C9:81	—
ECM dispatch from other	C8:78	—
ECM dispatch to DDR	C4:86	—
ECM dispatch to L2	C5:82	—
ECM dispatch to SRAM	C1:79	—
ECM dispatch to LBC	C6:82	—
ECM dispatch to PCI	C8:79	—
ECM dispatch to PEX2	C9:83	—
ECM dispatch to PEX1	C1:85	—
ECM dispatch to PEX3	C2:105	—

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM dispatch to CCSR	C2:84	—
ECM dispatch write	C1:81	—
ECM dispatch write allocate	C2:85	—
ECM dispatch read	C4:87	—
ECM dispatch read atomic clr, set, dec, inc	C9:87	—
ECM data bus grant DDR	C1:82	—
ECM data bus grant PCI/PEX/DMA	C2:86	—
ECM data bus grant I <sup>2</sup> C/Security	C3:86	—
ECM data bus grant LBC	C1:83	—
ECM data bus grant eTSEC1	C2:87	—
ECM data bus grant eTSEC3	C4:89	—
ECM global data bus beat	Ref:16	—
ECM e500 direct read bus beat	Ref:17	—
ECM e500 direct read bus beat forwarded	C2:88	ECM direct read bus beat forwarded directly to e500 R1 data bus
ECM cancel	Ref:18	—
Security/boot sequencer requests	C1:119	Number of sequencer requests (total)
Security/boot sequencer read requests	C2:76	Number of sequencer read requests
Security/boot sequencer data beats	C3:75	Number of sequencer data beats (total)
Security/boot sequencer read data beats	C4:79	Number of sequencer read data beats
Security/boot sequencer request less than 32 bytes	C7:76	Number of sequencer requests less than 32 bytes
<b>Interrupt Controller (PIC) Events</b>		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt wait cycles	C8:126	Counts cycles when an interrupt waits to be acknowledge
PIC interrupt service cycles	C2:83	Number of cycles there is an interrupt currently being serviced.
PIC interrupt select 0 (duration threshold)	C1:120	THRESHOLD: select 0–3: interrupt count over threshold. (Note: only unmasked, nonzero priority requests are acknowledged). The four interrupts are selected through register pairs, PM0MR <sub>n</sub> –PM3MR <sub>n</sub> . See <a href="#">Section 10.3.4, “Performance Monitor Mask Registers (PMMRs)”</a>
PIC interrupt select 1 (duration threshold)	C3:123	
PIC interrupt select 2 (duration threshold)	C5:119	
PIC interrupt select 3 (duration threshold)	C6:124	

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
<b>PCI Events</b>		
PCI clock cycles	Ref:28	—
PCI inbound memory reads	C1:126	Includes all read types.
PCI inbound memory writes	C2:101	—
PCI inbound config reads	C3:127	—
PCI inbound config writes	C4:101	—
PCI outbound memory reads	C5:94	Includes all read types.
PCI outbound memory writes	C6:96	Number of PCI outbound memory writes or number of PCI-X outbound memory writes attempted.
PCI outbound I/O reads	C3:101	—
PCI outbound I/O writes	C4:102	—
PCI outbound config reads	C7:90	Number of PCI outbound config reads OR number of PCI-X outbound config reads attempted.
PCI outbound config writes	C8:90	—
PCI inbound 32-bit read data beats	C1:94	—
PCI inbound 32-bit write data beats	C2:102	—
PCI outbound 32-bit read data beats	C3:102	—
PCI outbound 32-bit write data beats	C4:103	—
PCI total transactions	C7:93	Includes 32- and 64-bit transactions.
PCI inbound purgeable reads	C2:66	—
PCI inbound (speculative reads) purgeable reads discarded	C8:127	—
PCI idle cycles	C1:95	—
PCI dual address cycles	C2:104	—
PCI internal cycles	C3:103	—
PCI inbound memory read	C1:98	—
PCI inbound memory read line	C2:108	—
PCI inbound memory read multiple	C3:106	—
PCI outbound memory reads	C4:107	Number of PCI outbound memory reads or number of PCI-X outbound memory DWORD reads attempted.
PCI outbound memory read lines	C5:100	Number of PCI outbound memory read lines or number of PCI-X outbound memory burst reads attempted.
PCI wait	C1:99	PCI_IRDY, PCI_TRDY not both asserted
PCI snoopable	C1:96	—
PCI write stash	C2:106	—

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PCI write stash with lock	C3:105	—
PCI read unlock	C4:106	—
PCI byte enable transactions	C1:97	—
PCI non-byte enable transactions	C2:107	—
<b>eTSEC 1 Events</b>		
DMA write data beats	C3:109	DMA write data beats
DMA read data beats	C4:110	DMA read data beats
DMA Write Request	C5:106	DMA Write Request
DMA Read Request	C6:109	DMA Read Request
Number of dropped frames	C9:88	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:34	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:38	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:42	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:46	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:50	Read data lifetime
Rx IP packets checked for checksum	C9:92	Rx IP packets checked for checksum
TX IP packet with checksum	C1:105	TX IP packet with checksum
TX TCP/UDP packet with checksum	C2:113	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C3:114	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C4:115	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C5:111	Number of rejected frames by filer
Number of rejected frames due to filer error	C6:114	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C5:110	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C6:113	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C7:110	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C8:110	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:89	—
Number of accepted frames to station address	C8:101	—

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Number of accepted unicast frames via hash	C7:96	—
Number of accepted group frames via hash	C6:89	—
Number of accepted frames via exact match	C5:85	—
Number of rejected frames at layer 2	C4:105	—
Number of RX interrupts signalled	C3:82	—
Number of TX interrupts signalled	C2:110	—
RX data write lifetime (Duration Threshold)	Ref:43	—
Number of RX packets received while RX FIFO is full	Ref:47	—
<b>eTSEC 3 Events</b>		
DMA write data beats	C7:108	DMA write data beats
DMA read data beats	C8:108	DMA read data beats
DMA Write Request	C9:90	DMA Write Request
DMA Read Request	C1:103	DMA Read Request
Number of dropped frames	C4:112	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:36	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:40	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:44	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:48	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:52	Read data lifetime
Rx IP packets checked for checksum	C6:116	Rx IP packets checked for checksum
TX IP packet with checksum	C7:112	TX IP packet with checksum
TX TCP/UDP packet with checksum	C8:112	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C9:94	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C1:106	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C2:114	Number of rejected frames by filer
Number of rejected frames due to filer error	C3:116	Number of rejected frames due to filer error

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Number of cycles Rx FIFO > 1/4 full	C9:97	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C1:109	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C2:116	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C3:118	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:91	—
Number of accepted frames to station address	C8:102	—
Number of accepted unicast frames via has	C7:100	—
Number of accepted group frames via hash	C6:90	—
Number of accepted farms via exact match	C5:95	—
Number of rejected frames at layer 2	C4:111	—
Number of RX interrupts signalled	C3:85	—
Number of TX interrupts signalled	C2:112	—
RX data write lifetime (Duration Threshold)	Ref:45	—
Number of RX packets received while RX FIFO is full	Ref:49	—
<b>PCI Express 1 to OCeaN Gasket Events</b>		
Inbound G2PI read	C8:119	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C9:101	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:124	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:126	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:125	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:124	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:54	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:55	Lifetime of OSQ entry 0.
<b>PCI Express 2 to OCeaN Gasket Events</b>		
Inbound G2PI read	C8:70	A single pulse to indicate an inbound G2PI read has occurred.

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Inbound G2PI write	C9:64	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:74	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:74	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:73	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:72	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:56	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:57	Lifetime of OSQ entry 0.
<b>PCI Express 3 to OCeAN Gasket Events</b>		
Inbound G2PI read	C8:71	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C9:65	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:75	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:75	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:83	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:73	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:58	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:59	Lifetime of OSQ entry 0.
<b>Protocol Converter 1 Events</b>		
PEX Symbol Disparity Error	C8:117	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:99	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
<b>Protocol Converter 2 Events</b>		
PEX Symbol Disparity Error	C8:91	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:70	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
<b>Protocol Converter 3 Events</b>		

Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PEX Symbol Disparity Error	C8:94	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:71	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
<b>Local Bus Events</b>		
Bank 1 hits (chip-select)	C1:115	—
Bank 2 hits (chip-select)	C2:120	—
Bank 3 hits (chip-select)	C3:119	—
Bank 4 hits (chip-select)	C4:118	—
Bank 5 hits (chip-select)	C5:112	—
Bank 6 hits (chip-select)	C6:117	—
Bank 7 hits (chip-select)	C7:114	—
Bank 8 hits (chip-select)	C8:114	—
Requests granted to ECM port	C2:121	—
Cycles atomic reservation for ECM port is enabled	C4:119	—
Atomic reservation time-outs for ECM port	C6:118	—
Cycles a read is taking in GPCM	C1:117	—
Cycles a read is taking in UPM	C2:122	—
Cycles a read is taking in SDRAM	C3:121	—
Cycles a write is taking in GPCM	C4:120	—
Cycles a write is taking in UPM	C5:114	—
Cycles a write is taking in SDRAM	C6:119	—
SDRAM bank misses	C7:115	—
SDRAM page misses	C8:115	—
<b>L2 Cache/SRAM Events</b>		
Core instruction accesses to L2 that hit	Ref:22	—
Core instruction accesses to L2 that miss	C2:123	—
Core data accesses to L2 that hit	Ref:23	—
Core data accesses to L2 that miss	C4:121	—
Non-core burst write to L2 (cache external write or SRAM)	C5:115	—
Non-core non-burst write to L2	C6:120	—



Table 20-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Noncore write misses cache external write window and SRAM memory range	C7:116	—
Non-core read hit in L2	Ref:24	—
Non-core read miss in L2	C1:118	—
L2 allocates, from any source	Ref:25	—
L2 retries due to full write queue	C2:124	—
L2 retries due to address collision	C3:122	—
L2 failed lock attempts due to full set	C4:122	—
L2 victimizations of valid lines	C5:116	—
L2 invalidations of lines	C6:121	—
L2 clearing of locks	C7:117	—
<b>Debug Events</b>		
External event	C3:125	Number of cycles trig_in pin is asserted
Watchpoint monitor hits	C2:125	—
Trace buffer hits	C1:122	—
<b>DUART Events</b>		
UART0 baud rate	C1:127	—
UART1 baud rate	C5:127	—
<b>Chaining Events</b>		
PMC0 carry-out	Ref:1	PMC0[0] 1-to-0 transitions.
PMC1 carry-out	Ref:2	PMC1[0] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[0] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[0] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[0] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[0] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[0] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[0] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[0] 1-to-0 transitions. Reserved for PMC8.
PMC9 carry-out	Ref:10	PMC9[0] 1-to-0 transitions. Reserved for PMC9.

## 20.4.8 Performance Monitor Examples

Table 20-12 contains sample register settings for the four supported modes.

- Simple event performance monitoring example
- Triggering event performance monitoring example
- Threshold event performance monitoring example
- Burstiness event performance monitoring example

The settings in Table 20-11 are identical for all four examples.

**Table 20-11. PMGC0 and PMLCAn Settings**

Field	Setting	Reason
PMGC0[FAC]	0	Counters must not be frozen.
PMGC0[PMIE]	1	Performance monitor interrupts are enabled
PMGC0[FCECE]	1	Counters should be frozen when an interrupt is signalled.
PMLCAn[FC]	0	Counters cannot be frozen for counting.
PMLCAn[CE]	1	Overflow condition enable is required to allow interrupt signalling.

For simple event counting, a non-threshold event is selected in PMLCAn[EVENT] and all other features are disabled by clearing all register fields except for CE.

For the triggering example any event can be selected in PMLCAn[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signalling. If PMLCBn[TRIGONSEL] is 3 and PMLCBn[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCBn[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCBn[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCAn[CE] for this counter must be cleared so that interrupt signalling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCAn[EVENT]. For this example, the duration threshold value is scaled by two because PMLCBn[TBMULT] is one. All other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values for PMLCAn[Bsize,BGRAN,BDIST] and PMLCBn[TBMULT] must be specified.

**Table 20-12. Register Settings for Counting Examples**

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMGC0	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1

Table 20-12. Register Settings for Counting Examples (continued)

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMLCAn	FC	0	0	0	0
	CE	1	1	1	1
	EVENT	89	68	39	2
	BSIZE	0	0	0	5
	BGRAN	0	0	0	1
	BDIST	0	0	0	8
PMLCBn	TRIGONSEL	0	3	0	0
	TRIGOFFSEL	0	5	0	0
	TRIGONCNTL	0	1	0	0
	TRIGOFFCNTL	0	2	0	0
	TBMULT	0	0	0	0
	THRESHOLD	0	0	3	0

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.¶

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.



# Chapter 21

## Debug Features and Watchpoint Facility

This chapter describes all customer-visible debug modes of the MPC8533E integrated device. The device's debug features pertain to these interfaces: the local bus controller (LBC), and the DDR SDRAM interface. In addition to the external interfaces, the MPC8533E provides triggering capabilities based on user-programmable events. The watchpoint and trace buffer also provide some visibility to internal buses. This chapter also describes context ID registers, useful for software debug, and describes the JTAG access port signals that comply with the IEEE 1149.1 boundary-scan specification.

### 21.1 Introduction

As shown in the block diagram of [Figure 21-1](#), the device provides the following debug features (listed with references to sections of this chapter that describe them):

- DDR SDRAM interface debug ([Section 21.4.2, “DDR SDRAM Interface Debug”](#))
- Local bus controller (LBC) debug ([Section 21.4.3, “Local Bus Interface Debug”](#))
- Watchpoint monitor and trace buffer debug ([Section 21.4.4, “Watchpoint Monitor,”](#) and [Section 21.4.5, “Trace Buffer”](#))

#### 21.1.1 Overview

As shown in [Figure 21-1](#), debug information is provided through the following interfaces: LBC, and DDR SDRAM. Limited visibility, through a 256 x 64 trace buffer, is also provided for the processor core interface. This visibility into internal device operation is useful for debugging application software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID[0:4]) and a data-valid signal (MDVAL) indicates that meaningful debug information is visible on either the local bus or DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID[0:4] and MDVAL.

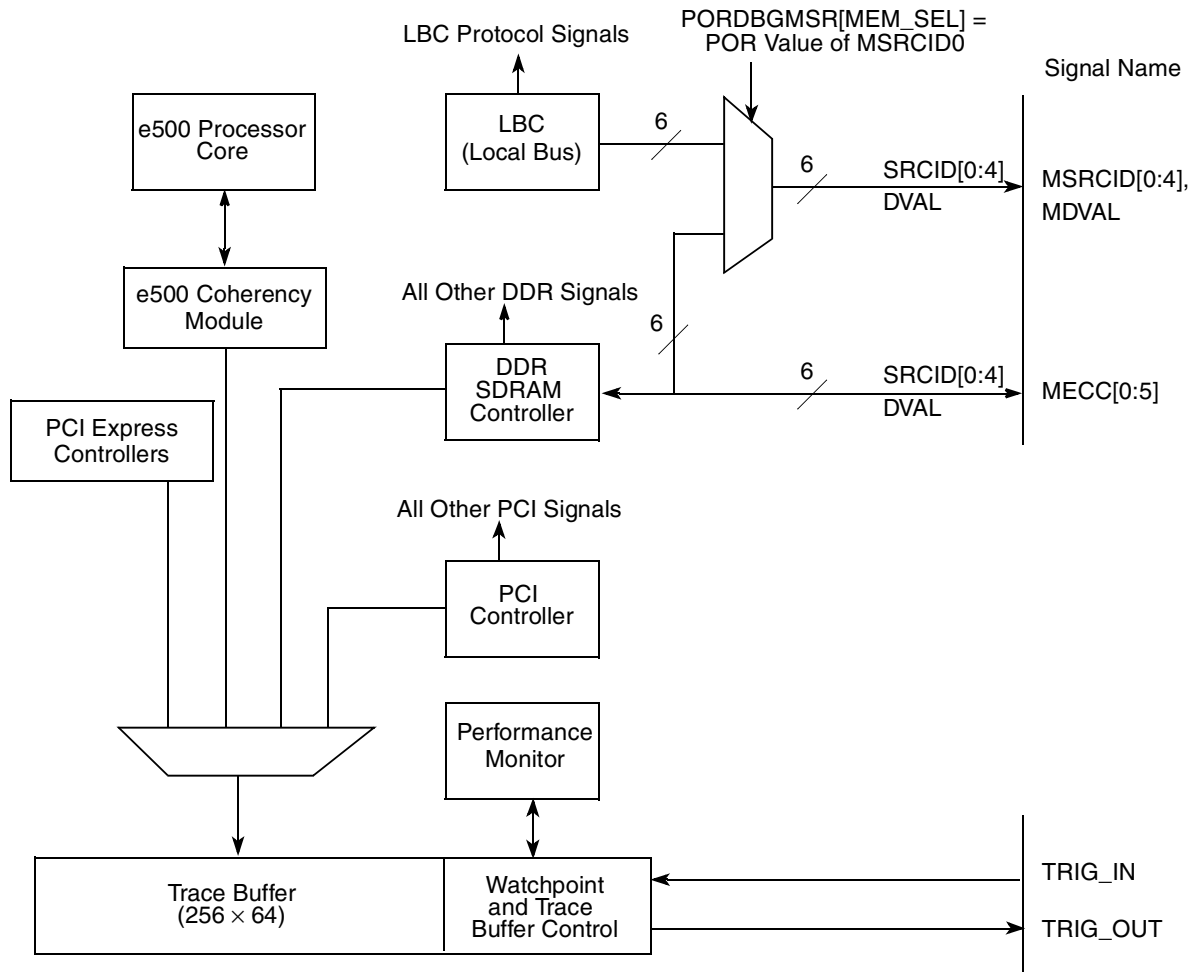


Figure 21-1. Debug and Watchpoint Monitor Block Diagram

Other system debugging is supported by the programmable triggering of the watchpoint monitor and trace buffer. Both can be triggered from one of the following three sources:

- Each other
- A performance monitor event
- An external source (through TRIG\_IN).

The watchpoint monitor can be configured to assert TRIG\_OUT when a programmed event occurs. The two context ID registers, described in [Section 21.3.3, “Context ID Registers,”](#) are useful for software debug.

## 21.1.2 Features

The principal features of the debug modes and the watchpoint monitor are as follows:

- LBC and DDR interface source ID and data-valid indicators
  - LBC or DDR SDRAM source ID can be selected to be driven onto MSRCID[0:4]
  - Source ID and data-valid indicators can be selected to be driven onto the error correcting code (ECC) pins of the DDR interface
- Watchpoint monitor that supports
  - Two-level triggering
  - Programmable external trigger (TRIG\_OUT)
  - Interlocked with performance monitor to use its large number of counters
- Trace buffer features that support
  - Two-level triggering
  - Programmable external trigger (TRIG\_OUT)
  - Interlocked with performance monitor to use its large number of counters
  - 256-entry trace buffer, 64 bits each
  - Programmable trace start and stop
  - Can function as a second watchpoint monitor
- Context ID registers that can be programmed to trigger events

## 21.1.3 Modes of Operation

The LBC, and DDR SDRAM interfaces all have debug modes, which are controlled by values on configuration inputs during the power-on reset (POR) sequence, as shown in [Table 21-1](#). The DDR controller can also drive debug information on either MSRCID[0:4] or MECC[0:5]. See [Section 21.4.1, “Source and Target ID,”](#) for additional information about the source ID information driven on the debug signals in these modes.

Note that both the watchpoint monitor and trace buffer also operate in a variety of modes.

**Table 21-1. POR Configuration Settings and Debug Modes**

Configuration Signal	POR Value	Effect	Reference
MSRCID0	0	Local bus SDRAM information appears on MSRCID[0:4] and MDVAL.	<a href="#">21.1.3.1/21-4</a>
	1	Default value (internal pull-up resistor). DDR SDRAM information appears on MSRCID[0:4] and MDVAL.	
MSRCID1	0	MECC[0:4] operate in debug mode and provide memory debug source ID and MECC5 provides data-valid information.	<a href="#">21.1.3.2/21-4</a>
	1	Default value (internal pull-up resistor). MECC[0:4] operate in normal mode and provide DDR SDRAM error correcting code information.	

### 21.1.3.1 Local Bus (LBC) Debug Mode

The LBC and the DDR SDRAM controller can drive debug information (source ID and data-valid indicator) onto MSRCID[0:4] and MDVAL. As shown in [Table 21-1](#), the MSRCID0 value during POR controls multiplexing. If MSRCID0 is low when sampled during POR, the local bus SDRAM information appears on MSRCID[0:4] and MDVAL; otherwise, the DDR SDRAM debug information is presented.

### 21.1.3.2 DDR SDRAM Interface Debug Modes

MSRCID1 is sampled during POR to multiplex either ECC or debug information on the ECC pins of the DDR SDRAM interface. As shown in [Table 21-1](#), if MSRCID1 is low during POR, the ECC pins operate in debug mode and provide memory debug source ID and data-valid information. MSRCID1 must be pulled low during POR to use the ECC pins in debug mode. If MSRCID1 is unconnected, an internal pull-up resistor ensures the ECC pins always source DDR SDRAM error correcting code information as their default power-on reset configuration.

#### NOTE

If the DDR ECC pins are in debug mode (configured for debug during POR), ECC checking is disabled in the memory controller. In this case, MECC[0:4] do not provide ECC information and must not be connected to SDRAM devices.

### 21.1.3.3 Watchpoint Monitor Modes

The watchpoint monitor supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The watchpoint monitor triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The watchpoint monitor waits for a specific event before enabling (arming) the trigger logic. The monitor does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Assert TRIG\_OUT on hit—The debug block can be programmed to assert the TRIG\_OUT signal when a programmed watchpoint monitor event occurs. This signal can be used to trigger a logic analyzer.

### 21.1.3.4 Trace Buffer Modes

The trace buffer supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The trace buffer triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The trace buffer waits for a specific event before enabling (arming) the trigger logic. The trace buffer does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Specific interface selection—The trace buffer can be programmed to trace one of several internal interfaces.



- Specific event selection—The trace buffer can be programmed to trace on the occurrence of one or several concurrent events.
- Specific trace selection—To facilitate trace data filtering, the trace buffer can be configured to capture data under the following conditions:
  - On every cycle in which a valid transaction is present on the selected interface
  - Only when the programmed trace event is detected
- Programmable trace stop—The trace buffer may be programmed to stop tracing when a programmed stop-tracing event occurs or when the 256-entry buffer is full.

## 21.2 External Signal Description

This section provides information about all the external signals associated with the various MPC8533E debug functions.

As shown in [Table 21-1](#), the MPC8533E has several signals that are sampled during POR to determine the configuration of the phase-locked loop clock mode and the ROM, flash, and dynamic memory. See [Chapter 4, “Reset, Clocking, and Initialization.”](#)

To facilitate system testing, the MPC8533E provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes JTAG TAP signals.

### 21.2.1 Overview

All the signals associated with device debug features are summarized in [Table 21-2](#), listed with a reference to the page number of the section with more information. The detailed descriptions are contained in [Table 21-2](#). Some signals (the MECC bus for example) are additionally described in other chapters, but are described here also for completeness, with emphasis on their debugging utility.

**Table 21-2. Debug, Watchpoint and Test Signal Summary**

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
MDVAL	Memory data-valid	Debug	Selectable data-valid signal from either DDR SDRAM controller or LBC.	1	O	<a href="#">21-6</a>
MECC[0:7]	DDR error correcting code	DDR SDRAM	In debug mode, the high-order six bits carry debug information (transaction source ID and data-valid indication).	0x08	O <sup>1</sup>	<a href="#">21-7</a>
MSRCID[0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM controller or local bus controller.	Reset_cfg	O	<a href="#">21-7</a>
MSRCID[2:4]				111	O	<a href="#">21-7</a>
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer.	1	I	<a href="#">21-7</a>
TRIG_OUT	Trigger out	Debug	Can be used externally for triggering a logic analyzer. Additionally, it can be used for observing system ready indication. Functions are multiplexed onto this signal depending on TOSR[SEL] (see <a href="#">Table 21-25</a> ).	1	O	<a href="#">21-7</a>
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up.	1	I	<a href="#">21-8</a>

**Table 21-2. Debug, Watchpoint and Test Signal Summary (continued)**

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	1	I	21-8
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	Hi Z	O	21-8
TMS	Test mode select	Debug	Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	1	I	21-8
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously.	—	I	21-8
THERM[0:1]	Thermal resistor access	Test	These pins tie directly to an internal resistor whose value varies linearly with temperature.	—	I	21-8
$\overline{\text{TEST\_SEL}}$	Test select 1	Test	Factory test. Must be negated (pulled high) for normal operation.	—	I	21-8
$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory Test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.		I	21-8
L1_TSTCLK	Test	Test	Factory Test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.		I	21-8
L2_TSTCLK	Test	Test	Factory Test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.		I	21-8

<sup>1</sup> While these signals are normally bidirectional, when sourcing debug information they are output only.

## 21.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals

### 21.2.2.1 Debug Signals—Details

Table 21-3 describes all signals associated with device debug modes.

**Table 21-3. Debug Signals—Detailed Signal Descriptions**

Signal	I/O	Description
MDVAL	O	Memory data-valid. Indicates when valid data is available. May be used by a logic analyzer to capture the data on the data bus.
		<b>State Meaning</b> Asserted—Indicates that data is valid on the data bus during the current clock cycle. When the DDR SDRAM interface is selected to source information on MDVAL, this signal is valid for every cycle that data is driven or received on the DDR SDRAM interface. When the LBC is selected, this signal is valid for every cycle that data is driven or received on the local bus interface. The assertion of this signal may be used by a logic analyzer to capture data.
		<b>Timing</b> Asserted/Negated—Referenced to the selected interface, (DDR or local bus). Asserts when data is valid. Assertions are held for the duration of the transfer. Read data timing is similar to MA. Write data timing is similar to the output MDQ.

Table 21-3. Debug Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MECC[0:7]	O	Memory ECC. DDR error checking and correcting. The normally bidirectional operation of the memory ECC (MECC) bus is described in <a href="#">Section 9.5.11, “Error Checking and Correcting (ECC).”</a> This bus is used for debug functions when MSRCID1 is sampled low during POR. In debug mode, the high-order 5 bits (MECC[0:4]) may be used to provide the transaction source ID and MECC5 can be used as the data-valid indicator. In debug mode, MECC[0:5] is constantly driven with debug information and must be disconnected from the DDR memory’s ECC pins.	
		<b>State Meaning</b>	Asserted/Negated—In debug mode, MECC[0:5] is always driven. The source ID values appear during RAS and CAS cycles. A value of 0x1F (all ones) is driven during cycles other than RAS and CAS. The data-valid indicator appears when data is being received or driven on the pins.
		<b>Timing</b>	Driven every cycle in debug mode.
MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in <a href="#">Table 21-1</a> . Two of these signals serve as reset configuration input signals.	
		<b>State Meaning</b>	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than RAS and CAS. The encodings shown in <a href="#">Table 21-26</a> provide detailed information about a memory transaction.
		<b>Timing</b>	Driven every cycle in debug mode. Similar timing to MA.

### 21.2.2.2 Watchpoint Monitor Trigger Signals—Details

[Table 21-4](#) shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 21-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TRIG_IN	I	Trigger in. Can be used to trigger the watchpoint and trace buffers. Note this is an active-high (rising-edge triggered) signal.	
		<b>State Meaning</b>	Asserted—Indicates that a programmed/armed external event has been detected. Assertion may be used internally to trigger trace buffers and watchpoint mechanisms.
		<b>Timing</b>	Assertion/Negation—The MPC8533E interprets TRIG_IN as asserted on detection of the rising edge. It may occur at any time. Must remain asserted for at least 3 system clocks to be recognized internally.

**Table 21-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions (continued)**

Signal	I/O	Description
TRIG_OUT	O	Trigger out. Function determined by TOSR[SEL]. When TOSR[SEL] is non-zero, it can be used for triggering external devices, like a logic analyzer, with either the watchpoint monitor, the trace buffer, or the performance monitor as trigger sources. When TOSR[SEL] is cleared, TRIG_OUT is multiplexed with READY, which indicates the operational readiness of the device (running or in low-power or debug modes). See <a href="#">Chapter 4, “Reset, Clocking, and Initialization,”</a> and <a href="#">Chapter 19, “Global Utilities,”</a> for more details about reset, low-power, and debug states.
		<b>State Meaning</b> Asserted—When TOSR[SEL] is all zeros, serves as the READY signal, indicating that the device is not in a low-power or debug mode and that it has emerged from reset. SEL ≠ 0 indicates that a programmed trigger event has occurred. Negation—No final watchpoint match condition
		<b>Timing</b> Assertion may occur at any time. Remains asserted for at least 3 system clocks

### 21.2.2.3 Test Signals—Details

Table 21-5 shows detailed descriptions of the JTAG test signals.

**Table 21-5. JTAG Test and Other Signals—Detailed Signal Descriptions**

Signal	I/O	Description
TCK	I	JTAG test clock.
		<b>State Meaning</b> Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		<b>Timing</b> See IEEE 1149.1 standard for more details.
TDI	I	JTAG test data input.
		<b>State Meaning</b> Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		<b>Timing</b> See IEEE 1149.1 standard for more details.
TDO	O	JTAG test data output.
		<b>State Meaning</b> Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		<b>Timing</b> See IEEE 1149.1 standard for more details.
TMS	I	JTAG test mode select.
		<b>State Meaning</b> Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		<b>Timing</b> See IEEE 1149.1 standard for more details.

Table 21-5. JTAG Test and Other Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{TRST}}$	I	JTAG test reset.	
		<b>State Meaning</b>	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the MPC8533E. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation.
		<b>Timing</b>	See IEEE 1149.1 standard for more details.
$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.	
L1_TSTCLK	I	Used for factory test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.	
L2_TSTCLK	I	Used for factory test. Refer to the <i>MPC8533E Integrated Processor Hardware Specifications</i> for proper treatment.	
THERM[0:1]	I	These signals provide access to an internal resistor that has a value that varies linearly with temperature. The actual value for the resistor varies from device to device, but the linear relationship between temperature and resistance is consistent. See the <i>MPC8533E Integrated Processor Hardware Specifications</i> for more information on how to accurately measure the junction temperature of a device. Note that this thermal resistor is intended for engineering development only.	
$\overline{\text{TEST\_SEL}}$	I	Used for factory test. Should be negated (pulled high) for normal operation.	

## 21.3 Memory Map/Register Definition

Table 21-6 shows the device's memory-mapped debug and watchpoint registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 21-6. Debug and Watchpoint Monitor Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
<b>Watchpoint Monitor Registers</b>				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	21.3.1.1/21-10
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	21.3.1.1/21-10

**Table 21-6. Debug and Watchpoint Monitor Memory Map (continued)**

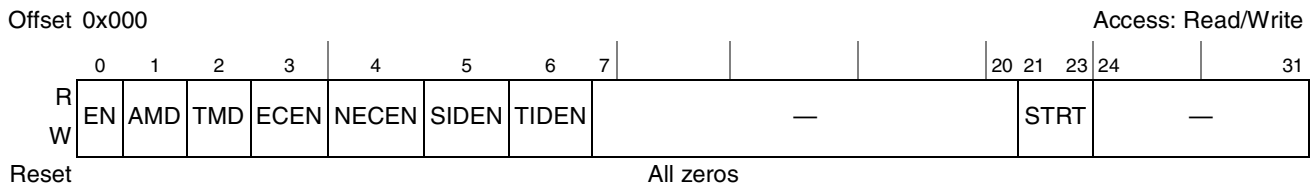
Local Memory Offset	Register	Access	Reset	Section/Page
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	21.3.1.2/21-12
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	21.3.1.3/21-13
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	21.3.1.4/21-13
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	21.3.1.5/21-15
<b>Trace Buffer Registers</b>				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	21.3.2.1/21-15
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	21.3.2.1/21-15
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	21.3.2.2/21-18
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	21.3.2.3/21-18
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	21.3.2.4/21-19
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	21.3.2.5/21-19
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	21.3.2.6/21-20
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	21.3.2.7/21-21
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	21.3.2.8/21-21
<b>Context ID Registers</b>				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	21.3.3.1/21-22
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	21.3.3.2/21-23
<b>Other Registers</b>				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	21.3.4.1/21-23

### 21.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

#### 21.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1) shown in [Figure 21-2](#) and [Figure 21-3](#) control the specification of watchpoint monitor events.



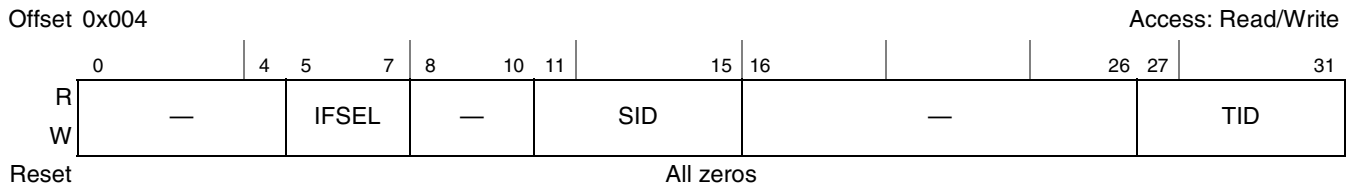
**Figure 21-2. Watchpoint Monitor Control Register 0 (WMCR0)**

Table 21-7 describes WMCR0 fields.

**Table 21-7. WMCR0 Field Descriptions**

Bits	Name	Description
0	EN	Enable 0 Watchpoint monitor events are not flagged. 1 A watchpoint monitor event is flagged.
1	AMD	Address match disable. Qualifies address match as a watchpoint event criterion. 0 Address matching is used to recognize a watchpoint event. 1 Address matching does not affect watchpoint event detection.
2	TMD	Transaction match disable. Qualifies transaction type match (as defined in WMCR1[IFSEL] and WMTMR) as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match does not affect watchpoint event detection.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in <a href="#">Section 21.3.3, "Context ID Registers."</a> 0 Current context match does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparing current context with the programmed context event value. <b>Note:</b> ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in <a href="#">Section 21.3.3, "Context ID Registers."</a> 0 The failure of a current context match does not affect watchpoint event detection 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. <b>Note:</b> ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Source ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 Target ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the watchpoint monitor to start looking for the programmed event. 000 No event. Armed immediately 001 Trace buffer event is detected 010 Performance monitor signals overflow 011 TRIG_IN transitions from 0 to 1 100 TRIG_IN transitions from 1 to 0 101 Current context ID equals programmed context ID 110 Current context ID is not equal to programmed context ID 111 Reserved
24–31	—	Reserved

Figure 21-3 shows WMCR1.



**Figure 21-3. Watchpoint Monitor Control Register 1 (WMCR1)**

Table 21-8 describes the WMCR1 fields.

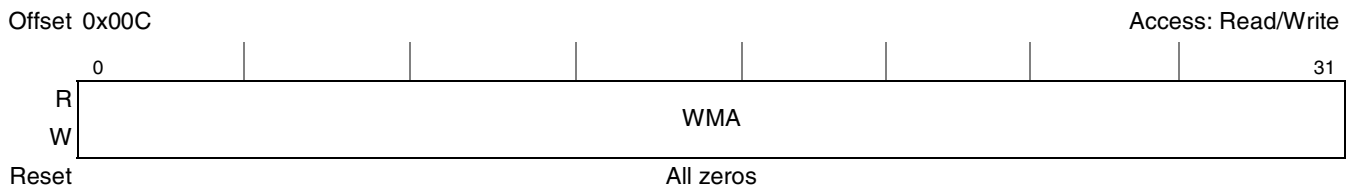
**Table 21-8. WMCR1 Field Descriptions**

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Selects the address, transaction type (as defined in WMTMR), and other attributes to be used for comparison 000 Selects e500 coherency module (ECM) dispatch interface 001 Selects internal DDR SDRAM interface 010 Selects internal PCI outbound interface 011 Reserved 100 Selects internal PCI Express 1 outbound interface 101 Selects internal PCI Express 2 outbound interface 110 Selects internal PCI Express 3 outbound interface 111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with WMCR0[SIDEN]. For a definition of the source ID, see <a href="#">Table 21-26</a> .
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with WMCR0[TIDEN]. For a definition of the target ID see <a href="#">Table 21-26</a> .

### 21.3.1.2 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR) shown in [Figure 21-4](#) contains the address to match against if WMCR[AMD] is clear. Note that this address may be further qualified with the bits described in [Section 21.3.1.3, “Watchpoint Monitor Address Mask Register \(WMAMR\).”](#)

Table 21-9 describes the WMAR fields.



**Figure 21-4. Watchpoint Monitor Address Register (WMAR)**

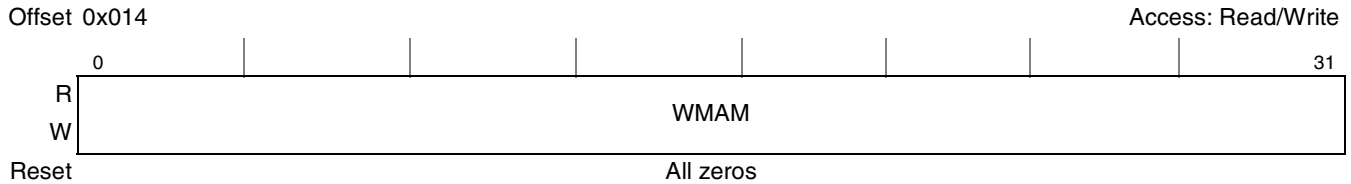


**Table 21-9. WMAR Field Descriptions**

Bits	Name	Description
0–31	WMA	Watchpoint monitor address.

### 21.3.1.3 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in [Figure 21-5](#) contains the mask for the address in the WMAR.



**Figure 21-5. Watchpoint Monitor Address Mask Register (WMAMR)**

[Table 21-10](#) describes the WMAMR fields.

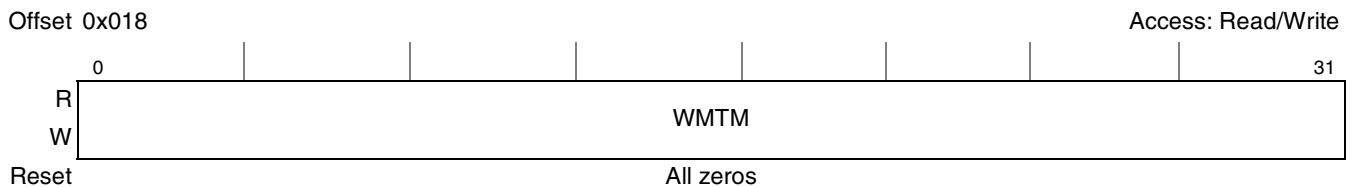
**Table 21-10. WMAMR Field Descriptions**

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. A value of zero masks the address comparison for the corresponding address bit. These bits only mask the address bits generated by the hardware, but do not affect the bits specified in WMAR. A bit that is masked from the comparison should be set to 0 in WMAR.

### 21.3.1.4 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in [Figure 21-6](#), specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in [Table 21-11](#), each bit represents as many as four separate transaction types; one for each interface. Setting a bit enables watchpoint monitoring for the corresponding transaction types.

Because the supported transaction types vary by interface, the type designated by a WMTMR field also depends on the interface specified by WMCRI[IFSEL]. [Table 21-12](#) lists transaction types associated with each WMTMR bit by interface.



**Figure 21-6. Watchpoint Monitor Transaction Mask Register (WMTMR)**

Table 21-11 describes the WMTMR fields.

**Table 21-11. WMTMR Field Descriptions**

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 21-12. The transaction associated with any particular bit may be different depending on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when WMCR0[TMD]=0.

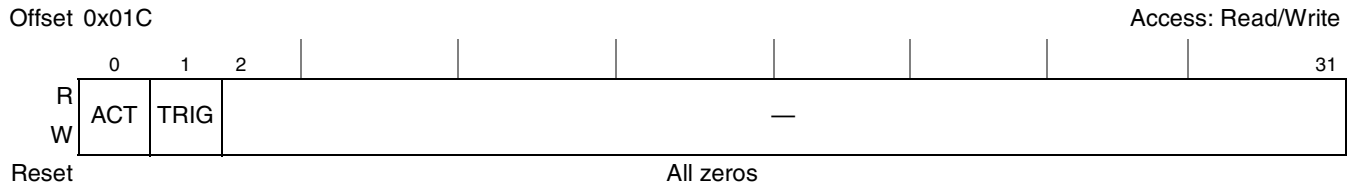
The following table, Table 21-12, defines the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

**Table 21-12. Transaction Types by Interface**

Bit	Description			
	e500 Coherency Module Dispatch	DDR Controller	PCI Outbound Request	PCI Express Outbound Transaction
0	Write with local processor snoop	Write	Memory write	Posted Write
1	Write with no local processor snoop	—	I/O write	Non-posted Write
2	Write with allocate(L2 stashing)	Write with allocate	—	—
3	Write with allocate and lock (L2 stashing with locking)	Write with allocate and lock	—	—
4	Reserved	—	—	—
5	Reserved	—	—	—
6	Reserved	—	—	—
7	Reserved	—	—	—
8	Read with local processor snoop	Read	Memory Read	Read
9	Read with no local processor snoop	—	I/O Read	—
10	Read with unlock	Read with unlock	—	—
11	Reserved	—	—	Read Response
12	Reserved	—	—	—
13–15	Reserved	—	—	—
16	ATOMIC clear	ATOMIC clear	—	—
17	ATOMIC set	ATOMIC set	—	—
18	ATOMIC decrement	ATOMIC decrement	—	—
19	ATOMIC increment	ATOMIC increment	—	—
20–24	Reserved	—	—	—
25	Address only transaction	—	—	—
26–31	Reserved	—	—	—

### 21.3.1.5 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR) shown in Figure 21-7 indicates the state of the watchpoint monitor.



**Figure 21-7. Watchpoint Monitor Status Register (WMSR)**

Table 21-13 describes the WMSR fields.

**Table 21-13. WMSR Field Descriptions**

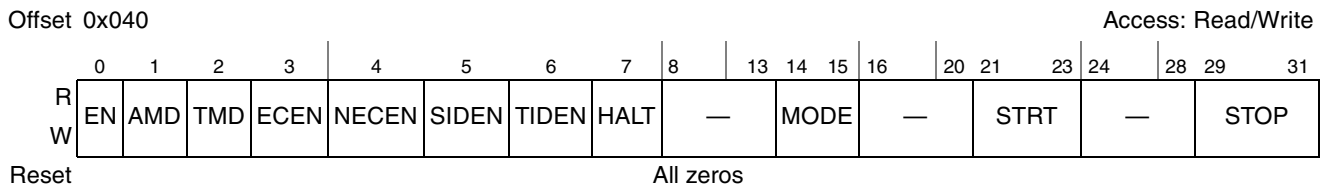
Bits	Name	Description
0	ACT	Active 0 The start triggering event has not occurred; watchpoint monitor is not armed. 1 The start triggering event has occurred; watchpoint monitor is armed.
1	TRIG	Triggered 0 The programmed event in WMCR0 has not yet been triggered. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

### 21.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

#### 21.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 21-8 and Figure 21-9, specify trace buffer events.



**Figure 21-8. Trace Buffer Control Register 0 (TBCR0)**

Table 21-14 describes the TBCR0 fields.

**Table 21-14. TBCR0 Field Descriptions**

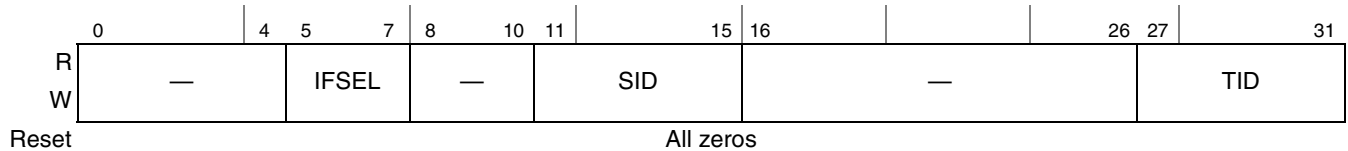
Bits	Name	Description
0	EN	Enable 0 The trace buffer facility is disabled. 1 The trace buffer facility is enabled.
1	AMD	Address match disable 0 The address match is used to qualify a trace buffer event. 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 The transaction type match is used to qualify a trace buffer event. 1 The transaction type match is ignored when detecting a trace buffer event.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in <a href="#">Section 21.3.3, "Context ID Registers."</a> 0 Current context match does not affect trace buffer event detection 1 Trace buffer events are qualified by comparing current context with the programmed context event value. <b>Note:</b> ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in <a href="#">Section 21.3.3, "Context ID Registers."</a> 0 The failure of a current context match does not affect trace buffer event detection 1 trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. <b>Note:</b> ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Trace buffer events ignore the programmed source ID value. 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 Trace buffer events ignore the programmed TID event value. 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the ECM is selected for tracing (TBCR1[IFSEL] is all zeros).
7	HALT	Halt causes the trace buffer to stop tracing immediately. TBSR[ACT] remains set when this bit is set.
8–13	—	Reserved
14–15	MODE	Trace mode. Specifies one of two trace modes. 00 Trace every valid transaction 01 Reserved 10 Trace only cycles in which a trace event is detected. Note that if EN and other TBCR0 fields are not properly programmed to specify a traceable event, tracing occurs for every valid address. 11 Reserved
16–20	—	Reserved

**Table 21-14. TBCR0 Field Descriptions (continued)**

Bits	Name	Description
21–23	STRT	Start condition. Specifies the event that arms the trace buffer to start looking for the programmed event 000 No event. Armed immediately 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programmed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. Specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stops after it has been triggered at least once. 000 Buffer is full 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programmed context ID

Offset 0x044

Access: Read/Write



**Figure 21-9. Trace Buffer Control Register 1 (TBCR1)**

Table 21-15 describes the TBCR1 fields.

**Table 21-15. TBCR1 Field Descriptions**

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Specifies the interface that sources information for both comparison/buffer control and buffer data capture. 000 Selects e500 coherency module (ECM) dispatch interface 001 Selects internal DDR SDRAM interface 010 Selects internal PCI outbound interface 011 Reserved 100 Selects internal PCI Express 1 outbound interface 101 Selects internal PCI Express 2 outbound interface 110 Selects internal PCI Express 3 outbound interface 111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with TBCR0[SIDEN]. The source ID is defined in Table 21-26.

**Table 21-15. TBCR1 Field Descriptions (continued)**

Bits	Name	Description
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in <a href="#">Table 21-26</a> .

### 21.3.2.2 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR) shown in [Figure 21-10](#) contains the address to match against (if TBCR0[AMD] is zero). This address may be further qualified by the mask bits defined in [Section 21.3.2.3](#), “Trace Buffer Address Mask Register (TBAMR).”



**Figure 21-10. Trace Buffer Address Register (TBAR)**

[Table 21-16](#) describes the TBAR field.

**Table 21-16. TBAR Field Descriptions**

Bits	Name	Description
0–31	TBA	Trace buffer address.

### 21.3.2.3 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR) shown in [Figure 21-11](#) contains a mask for the TBAR, which allows excluding address bits from the comparison.



**Figure 21-11. Trace Buffer Address Mask Register (TBAMR)**

[Table 21-17](#) describes the TBAMR field.

**Table 21-17. TBAMR Field Descriptions**

Bits	Name	Description
0–31	TBAM	Trace buffer address mask. A value of zero masks the address comparison for the corresponding address bit. These bits only mask the address bits generated by the hardware, but do not affect the bits specified in TBAR. A bit that is masked from the comparison should be set to 0 in TBAR.

### 21.3.2.4 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR) shown in Figure 21-12 specifies which transaction types to monitor. Each bit in the TBTMR represents a transaction type on the selected interface. The transaction associated with any particular bit depends on the interface being monitored as specified by TBCR1[IFSEL]. Note that the transactions used for defining trace buffer events are the same as those defined for watchpoint monitor events. Thus, Table 21-12 defines the transaction types associated with each interface. Setting a bit enables a hit when this transaction is matched (provided all other match criteria are met and TBCR0[TMD] is clear).

Different interfaces support different transaction types, and the same bit may represent different transaction types depending on the interface.

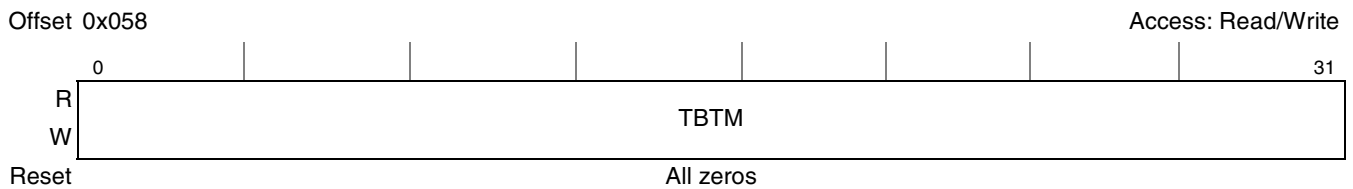


Figure 21-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 21-18 describes the TBTMR field.

Table 21-18. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTMR	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 21-12. The transaction associated with a bit depends on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when TBCR0[TMD]=0.

### 21.3.2.5 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR) shown in Figure 21-13 indicates the operational state of the trace buffer.

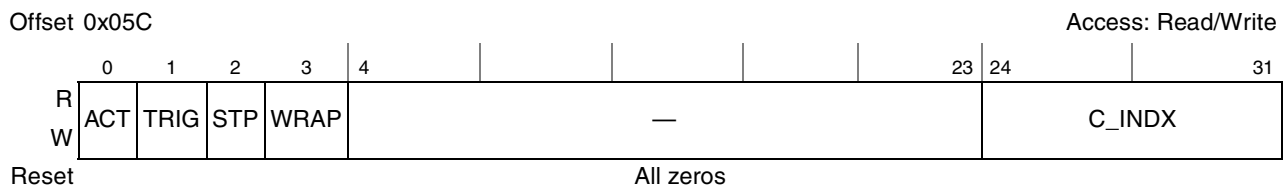


Figure 21-13. Trace Buffer Status Register (TBSR)

Table 21-19 describes the TBSR fields.

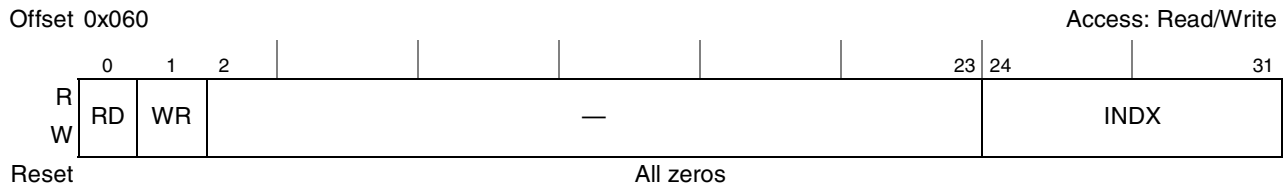
**Table 21-19. TBSR Field Descriptions**

Bits	Name	Description
0	ACT	Active. Indicates trace buffer activity. 0 The start triggering event has not yet occurred. Trace buffer is not armed. 1 The start triggering event has occurred. Trace buffer is armed.
1	TRIG	Triggered. Indicates whether or not a programmed event has been triggered. 0 The programmed event in TBCR0 has not yet been triggered. 1 The programmed event in TBCR0 has been triggered at least once.
2	STP	Stopped. Indicates whether or not a trace buffer stop condition has been detected. 0 No stop condition yet detected. 1 The trace buffer has detected a stop condition and is no longer capturing events.
3	WRAP	Wrapped. Indicates that the trace buffer write pointer has wrapped to the beginning of the buffer at least once. Set when the last entry of the trace buffer is written. 0 Pointer has not yet wrapped. 1 Pointer has wrapped to the beginning at least once.
4–23	—	Reserved
24–31	C_INDx	Current index. Represents the current value of the write pointer at the time TBSR was read. This value may be written by software to initialize the write pointer; however, software is not allowed to write the write pointer while the trace buffer is active. Writes are ignored while the trace buffer is active. It is recommended to write the status register before enabling the trace buffer in order to zero out any bits that might have been set during a prior run and to initialize the write pointer to zero.

### 21.3.2.6 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR) enables software to read or write the trace buffer. Each entry is 64 bits; therefore, it takes one write of TBACR and two reads of the access data register (TBADR and TBADHR) to read one 256-entry array entry. Similarly, it takes one write of TBACR and two writes of TBADR and TBADHR to write one array entry. Software can access any entry by writing the appropriate index into TBACR[INDX]. To read or write the buffer sequentially, starting with entry 0, the index must start with a value of 0 and increment every time a new entry is accessed.

TBACR is shown in Figure 21-14.



**Figure 21-14. Trace Buffer Access Control Register (TBACR)**



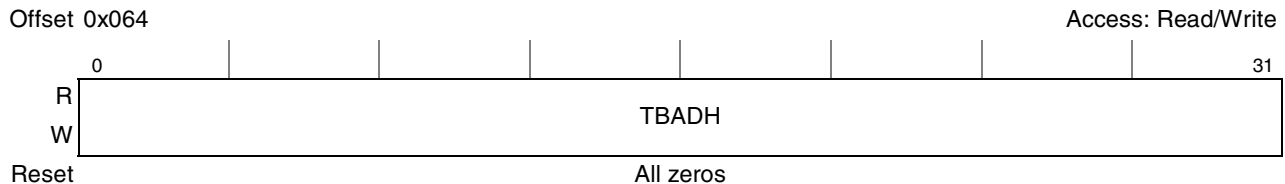
Table 21-20 describes the TBACR fields.

**Table 21-20. TBACR Field Descriptions**

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. This bit is automatically cleared when the read is performed.
1	WR	Write command. When set, a trace buffer write is performed using the value of TBACR[INDX]. This bit is automatically cleared when the write is performed. A write occurs only if the trace buffer is not active: write requests are ignored while the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer index to read from or write into (0–255). Used in conjunction with TBACR[RD] and TBACR[WR].

### 21.3.2.7 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 21-15, contains the high-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]), or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.



**Figure 21-15. Trace Buffer Read High Register (TBADHR)**

Table 21-21 describes TBADHR.

**Table 21-21. TBADHR Field Descriptions**

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. The higher 32 bits of the data read from or to be written into the trace buffer, depending on whether the array is accessed with a read or a write.

### 21.3.2.8 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 21-16, contains the low-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]) or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

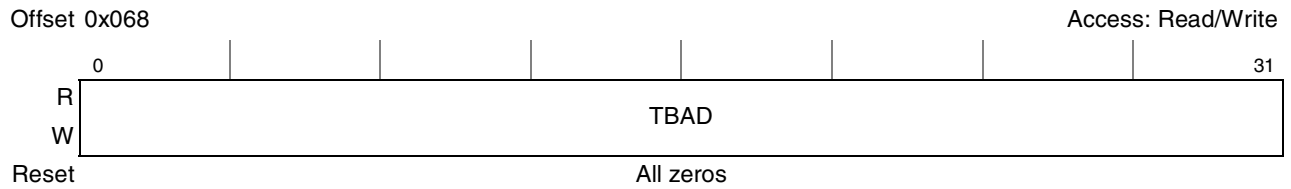


Figure 21-16. Trace Buffer Access Data Register (TBADR)

Table 21-22 describes the TBADR field.

Table 21-22. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. Corresponds to the lower 32 bits of the data read from the trace buffer or to be written into the trace buffer, depending on whether software is accessing the array with a read or a write.

### 21.3.3 Context ID Registers

This section describes the context ID registers. The current context ID register (CCIDR) and programmed context ID registers (PCIDR) are set by software and facilitate debugging complex software.

#### 21.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 21-17, contains the user-programmed context ID. This register can be configured to trigger watchpoint events when its value matches the current context ID register (CCIDR), as controlled by WMCR0[ECEN] and WMCR0[NECEN]. See Section 21.3.1.1, “Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1),” for more information.

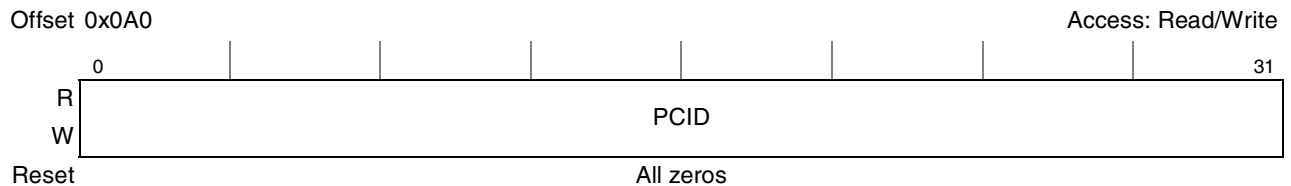


Figure 21-17. Programmed Context ID Register (PCIDR)

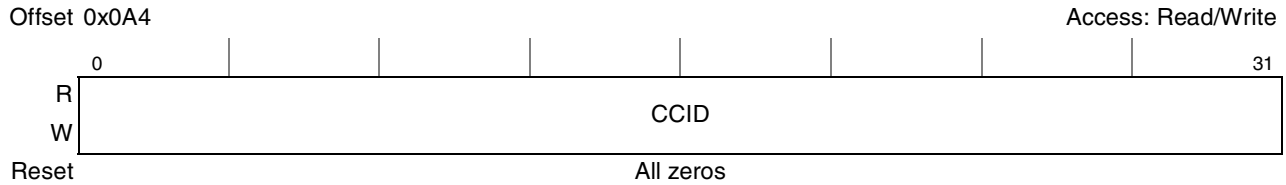
Table 21-23 describes the PCIDR field.

Table 21-23. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed context ID. Contains the user-programmed context ID. Compared with current context ID for context-sensitive event triggering

### 21.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR) shown in [Figure 21-18](#) contains the current context ID. This register is written by software after a context switch and can be used to trigger events when compared with the programmed context ID register (PCIDR).



**Figure 21-18. Current Context ID Register (CCIDR)**

[Table 21-24](#) describes the CCIDR field.

**Table 21-24. CCIDR Field Descriptions**

Bits	Name	Description
0–31	CCID	Current context ID. Set by user software. Typically loaded immediately following a context switch. Compared with user-programmed context ID for context-sensitive event triggering

### 21.3.4 Trigger Out Function

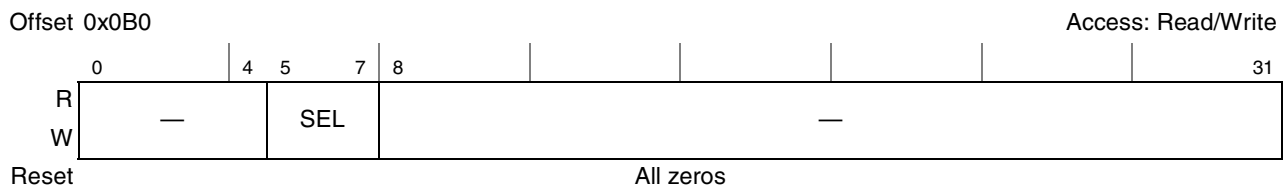
TRIG\_OUT provides a convenient mechanism for triggering external system monitors and diagnostic equipment such as logic analyzers. Note that READY is multiplexed with TRIG\_OUT. See the last paragraph of [Section 4.4.2, “Power-On Reset Sequence,”](#) for more information about READY functionality.

When the trace buffer hit is selected by TOSR[SEL], TRIG\_OUT is only meaningful if the trace buffer control register 0 (TBCR0) is properly configured to hit on a traceable event. The same holds true for the watchpoint monitor when the watchpoint monitor is selected by TOSR[SEL].

#### 21.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in [Figure 21-19](#) specifies the source for TRIG\_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor



**Figure 21-19. Trigger Out Source Register (TOSR)**

Table 21-25 describes the TOSR fields.

**Table 21-25. TOSR Field Descriptions**

Bits	Name	Description
0–4	—	Reserved
5–7	SEL	Select. Selects the source for TRIG_OUT 000 READY signal. Multiplexed with TRIG_OUT. Basic device state indicator. READY asserts whenever the device is not in reset or not asleep. See <a href="#">Chapter 4, “Reset, Clocking, and Initialization,”</a> for more details about the reset sequence, and <a href="#">Chapter 19, “Global Utilities,”</a> for more information about power management states. 001 Selects the watchpoint monitor hit indication 010 Selects the trace buffer hit indication 011 Selects the performance monitor overflow indication
8–31	—	Reserved

## 21.4 Functional Description

The debug features on the MPC8533E use the LBC interfaces, and the DDR SDRAM interface.

### 21.4.1 Source and Target ID

Debug information that is common to all the interfaces is the source ID (SID). The transaction source ID provides enough information to determine which block or port originated a transaction including the distinction between instruction and data fetches from the processor core. [Table 21-26](#) shows the values and interpretation for the 5-bit SID field. Note that the table also includes ports that are only slaves, such as local memory. These ports are always targets. As such, the value shown represents a target ID (TID) and not a source ID. For ports that can function in both capacities, the value indicates source ID when mastering transactions, and target ID when responding as slave. The TID field is only meaningful when one of the following participates in the transaction:

- The e500 coherency module (ECM) dispatch bus
- The watchpoint monitor (WMCR1[IFSEL] = 000)
- The trace buffer (TBCR1[IFSEL] = 000)

Table 21-26. Source and Target ID Values

Value (Hex)	Source (or Target) Port	Value (Hex)	Source (or Target) Port
00	PCI	10	Local processor (instruction fetch)
01	PCI Express 2	11	Local processor (data fetch)
02	PCI Express 1	12	Reserved
03	PCI Express 3	13	Reserved
04	Local bus controller	14	Reserved
05	Reserved	15	DMA
06	Reserved	16	Reserved
07	Security	17	System access port (SAP)
08	Configuration space	18	eTSEC1
09	Reserved	19	Reserved
0A	Boot sequencer	1A	eTSEC3
0B	Reserved	1B	Reserved
0C	Reserved	1C	Reserved
0D	Reserved	1D	Reserved
0E	Reserved	1E	Reserved
0F	Local space (DDR)	1F	Non valid port indicator (reserved for debug info)

## 21.4.2 DDR SDRAM Interface Debug

The DDR interface has two debug modes distinguished by which pins drive the debug information. In one mode, debug information (source ID, data valid) is multiplexed onto the ECC pins; the other mode uses the debug pins.

### 21.4.2.1 Debug Information on Debug Pins

If MSRCID0 is high when sampled during POR, the debug information from the DDR SDRAM interface is driven on MSRCID[0:4] and MDVAL. This POR value is captured in PORDBGMSR[MEM\_SEL] as described in [Section 19.4.1.5, “POR Debug Mode Status Register \(PORDBGMSR\).”](#) In this mode, the source ID appears on MSRCID[0:4] during a RAS or CAS cycle. During any other cycle, the value of MSRCID[0:4] is all ones, which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

### 21.4.2.2 Debug Information on ECC Pins

If MSRCID1 is low when sampled during POR, debug information from the DDR SDRAM interface is selected to appear on MECC[0:5] as shown in [Figure 21-1](#). In this mode, the ID value of the source port, (the source ID), appears on MECC[0:4] during a RAS or CAS cycle. During any other cycle the value of

MECC[0:4] is all ones. A data-valid signal (DVAL) is driven on MECC5 during valid DDR SDRAM data cycles.

### NOTE

In this mode, MECC[0:5] must be disconnected from all SDRAM devices to prevent contention on those lines.

## 21.4.3 Local Bus Interface Debug

If MSRCID0 is low when sampled during POR, the LBC is selected as the source for the debug information appearing on MSRCID[0:4] and MDVAL. For more information on this mode, see [Section 14.1.3.2, “Source ID Debug Mode.”](#)

## 21.4.4 Watchpoint Monitor

The watchpoint monitor (WM) can be programmed to arm and trigger on many different events including any of the following:

- External event (through TRIG\_IN)
- A trace buffer event
- A performance monitor overflow event
- A comparison of the current and programmed context ID registers

A watchpoint event can be used in the following ways:

- Trigger a logic analyzer (using TRIG\_OUT)
- Arm or trigger the trace buffer
- Trigger a performance monitor event

The large counters available in the performance monitor block and the interlock between it and the watchpoint monitor support sophisticated debug scenarios.

A WM trigger event may be composed of several events programmed in the watchpoint monitor control registers (WMCR0–WMCR1). Because the watchpoint monitor is disabled by default during POR, these registers must be initialized to make use of this debug feature. Note that the WM address mask register (WMAMR) and the type mask register (WMTMR) are cleared during POR. This means that the watchpoint monitor’s default behavior following a power-on reset is to trigger on any address and no transaction type. The reset value of WMCR0[TMD] is 0 which means transaction matching is enabled but since no transaction is selected (WMTMR=0), a match will never occur. Either the transaction matching must be disabled by setting WMCR0[TMD] to a value of 1, or valid transactions must be selected by setting one or more of the WMTMR bits to a value of 1.

### 21.4.4.1 Watchpoint Monitor Performance Monitor Events

The WM can produce a performance monitor (PM) event with every trigger. This is accomplished by configuring the performance monitor to count WM events. For more information on this configuration see the events named ‘Number of watchpoint monitor hits’ and ‘Number of trace buffer hits’ in [Table 20-10](#).

Multi-level triggers can be created using the watchpoint monitor, the performance monitor, and the trace buffer combined. For example, the WM can be programmed to trigger on events that also increment a PM counter (the performance monitor must also be programmed to respond to this event), the output of which (perfmon\_overflow) could trigger the start of tracing in the trace buffer.

### 21.4.5 Trace Buffer

The trace buffer is a 256 × 64 array that can capture information about the internal processing of transactions to selected interfaces. The trace buffer controls are a superset of those for the watchpoint monitor. Close inspection of the trace buffer control registers (TBCR<sub>n</sub>) and the WM control registers (WMCR<sub>n</sub>) shows that trace buffer controls not needed for the WM are marked reserved in WMCR<sub>n</sub>. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer provides great flexibility about when to start tracing, when to stop tracing, and what to trace. The trace mode field, TBCR0[MODE], indicates when to trace: on every valid cycle, on a watchpoint monitor event, or when all the programmed events in the TBCR are met. This permits a user to program the trace condition in the watchpoint monitor and to program a start or stop condition in the trace buffer control register. The user can also program the TBCR with the conditions in which to stop tracing: on an event, or when the buffer is full. TBCR0[IFSEL] specifies which interface transactions are being captured.

The trace buffer can be programmed to trace the dispatch bus from any of the following:

- e500 coherency module (ECM)
- Outbound host interface to the PCI controller
- Host interface to the DDR controller

Transactions come into the ECM, arbitrate for common resources, and get dispatched to the target port. Information such as transaction types, source ID, and other attributes can be captured in any of the selected interfaces.

#### 21.4.5.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 21-20 shows the trace buffer entry format for an ECM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

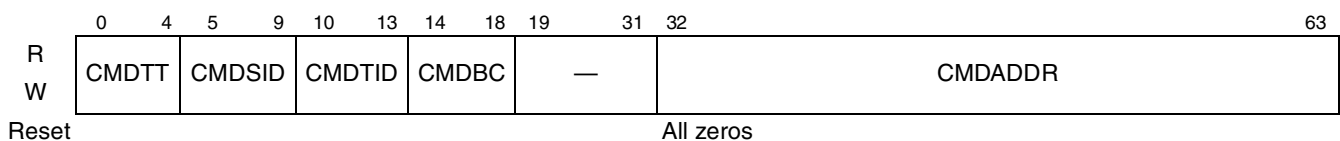


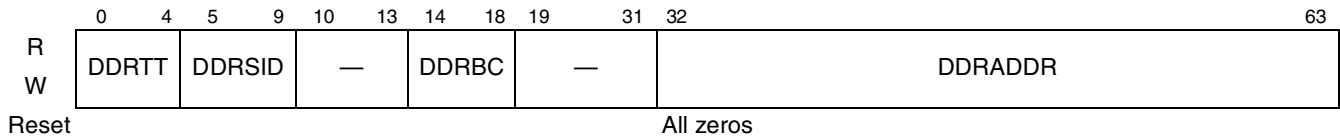
Figure 21-20. e500 Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 21-27 describes the fields of CMD trace buffer entries.

**Table 21-27. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)**

Bits	Name	Function
0–4	CMDTT	Transaction type. Specifies the transaction type as shown in Table 21-12. For example, a value of zero indicates a write with local processor snoop condition.
5–9	CMDSID	Source ID. Identifies the source of the transaction as shown in Table 21-26. For example, a value of 010101 indicates that DMA is the transaction source.
10–13	CMDTID	Target ID. Identifies the target of the transaction as shown in Table 21-26. For example, a value of 010101 indicates that DMA is the transaction target.
14–18	CMDBC	Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes. 00000 = 32 bytes 00001 = 1 byte 00010 = 2 bytes ... 11110 = 30 bytes 11111 = 31 bytes
19–31	—	Reserved
32–63	CMDADDR	Address bits 0–31

Figure 21-21 shows the trace buffer entry format for the DDR SDRAM interface, TBCR1[IFSEL] = 001.



**Figure 21-21. DDR Trace Buffer Entry**

Table 21-28 describes the fields of DDR SDRAM trace buffer entries when TBCR1[IFSEL] = 001.

**Table 21-28. DDR Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 001)**

Bits	Name	Function
0–4	DDRTT	Transaction type. Specifies the transaction type as shown in Table 21-12. For example, a value of all zeros maps to write.
5–9	DDRSID	Source ID. Specifies the source of the transaction as shown in Table 21-26. For example, a value of 010101 indicates that DMA is the transaction source, and so on.
10–13	—	Reserved
14–18	DDRBC	Byte count
19–31	—	Reserved
32–63	DDRADDR	Address bits 0–31



Figure 21-22 shows the PCI trace buffer entry format when TBCR1[IFSEL] = 010 or 101.

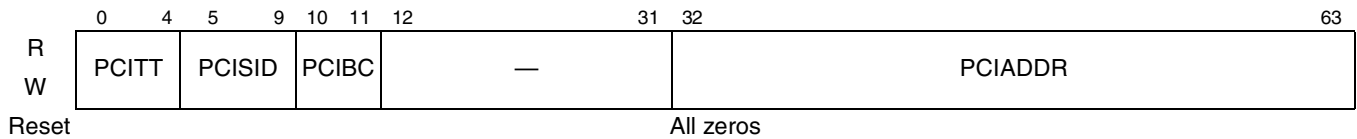


Figure 21-22. PCI Trace Buffer Entry

Table 21-29. PCI Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 010)

Bits	Name	Function
0–4	PCITT	Transaction type. Specifies the transaction type as shown in Table 21-12. For example, a value of all zeros maps to write.
5–9	PCISID	Source ID. Identifies the source of the transaction as shown in Table 21-26. For example, a value of 010101 identifies DMA as the transaction source.
10–11	PCIBC	Byte count. The size of the transaction. 00 32 bytes 01 8 bytes 10 16 bytes 11 24 bytes
12–31	—	Reserved
32–63	PCIADDR	Address bits 0–31

Figure 21-23 shows the PCI Express trace buffer entry format when TBCR1[IFSEL] = 100 or 101 or 110.

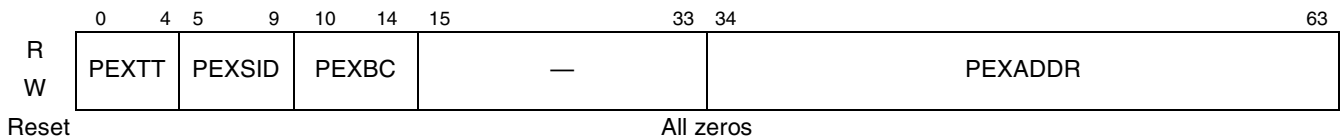


Figure 21-23. PCI Express Trace Buffer Entry

Table 21-30 describes the fields of PCI Express trace buffer entries when TBCR1[IFSEL] = 100 or 101 or 110.

Table 21-30. PCI Express Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 100 or 101 or 110)

Bits	Name	Function
0–4	PEXTT	Transaction type. Specifies the transaction type as shown in Table 21-12. For example, a value of all zeros maps to write.
5–9	PEXSID	Source ID. Identifies the source of the transaction as shown in Table 21-26. For example, a value of 010101 identifies DMA as the transaction source. For responses, this corresponds to Requestor's ID's bus number bits 3–7.

**Table 21-30. PCI Express Trace Buffer Entry Field Descriptions  
(TBCR1[IFSEL] = 100 or 101 or 110) (continued)**

Bits	Name	Function
10–14	PCIBC	Byte count. The size of the transaction. 00000 4 bytes 00001 8 bytes 00010 12 bytes ... 11111 256 bytes
15–33	—	Reserved
34–63	PEXADDR	Address bits 31–2

## 21.5 Initialization

Configuring the appropriate control register must be the last step in the initialization sequence for either the watchpoint or trace buffer. That is, all required registers except the corresponding control register must be configured before any control register bits that enable watchpoint or trace events are set.

# Appendix A

## Revision History

This appendix provides a list of major differences between revisions of the *MPC8533E PowerQUICC III Integrated Host Processor Family Reference Manual*.

### A.1 Changes From Revision 0 to Revision 1

Major changes to the *MPC8533E Integrated Processor Reference Manual*, from Revision 0 to Revision 1 are as follows:

Section/Page	Changes
<a href="#">3.1/3-1</a>	Added <a href="#">Table 3-2</a> .
<a href="#">5.2/5-4</a>	In <a href="#">Table 5-1</a> , changed PVR/SVR values.
<a href="#">Chapter 7/7-1</a>	Modified chapter to accommodate 256-Kbyte cache.
<a href="#">9.4.1.28/9-38</a>	In <a href="#">Table 9-34</a> , added to the bit field description of TSIZ: 000 4 double words 001 1 double word 010 2 double words 011 3 double words Others Reserved
<a href="#">11.3.1.2/11-6</a>	In <a href="#">Table 11-5</a> , corrected the bit field description of FDR from: “serial bit clock frequency of SCL is equal to one half the platform (CCB) clock” to “serial bit clock frequency of SCL is equal to one third the platform (CCB) clock”
<a href="#">11.3.1.4/11-9</a>	In <a href="#">Table 11-7</a> , corrected the bit field description of MCF. Specifically, changed “I2CSR” to “I2CDR” for the received case conditions.
<a href="#">11.3.1.6/11-11</a>	In <a href="#">Table 11-9</a> , corrected the bit field description of DFSR from: “sampling rate is calculated by dividing one half the platform (CCB clock) frequency” to “sampling rate is calculated by dividing one third the platform (CCB clock) frequency”
<a href="#">Chapter 15/15-1</a>	“eTSEC3” was sometimes referenced as “eTSEC2”; all references in the chapter to “eTSEC2” should be changed to “eTSEC3”.

## Revision History

15.4/15-6	In <a href="#">Table 15-1</a> , updated the descriptions of TXD[7:4] and RXD[7:4] to include MII in the function description.
15.4.1/15-8	In <a href="#">Table 15-2</a> , changed TSEC <sub>n</sub> _TX_EN signal description from “rising and falling edges of the TSEC <sub>n</sub> _TX_CLK, respectively.” to “rising and falling edges of the TSEC <sub>n</sub> _GTX_CLK, respectively.”
15.5.1/15-13	In <a href="#">Table 15-3</a> , added a row for the lossless flow control registers.
15.5.2/15-13	In <a href="#">Table 15-4</a> , added a section of lossless flow control registers.
15.5.3.2.3/15-42	Changed the fourth sentence in DFVLAN introductory text from: “Frames with an Ethertype of 0x8808 will be dropped by the receiver unless RCNTRL[CFA] is set.” to “Frames with an Ethertype of 0x8808 will be dropped by the receiver.”
15.5.3.3.1/15-48	In <a href="#">Figure 15-22</a> and <a href="#">Table 15-26</a> , added bit 17, LFC (lossless flow control). In addition, clarified the bit field description of RSF.
15.5.3.3.8/15-58	In <a href="#">Table 15-33</a> , clarified the bit field description of ETY with a description of parser/filer behavior with ethertype field (parsed into ETY). Also, added text to the bit field description of IPF.
15.5.3.5.2/15-68	In <a href="#">Table 15-40</a> , clarified that the bit field description of “Preamble Length” may be from 0x3 to 0xF.
15.5.3.6.26/15-92	In <a href="#">Table 15-80</a> , changed bit field TPKT to 22 bits (bits 10–31).
15.5.3.9.2/15-111	In <a href="#">Figure 15-104</a> and <a href="#">Table 15-107</a> , corrected the size and bit field description of EI.
15.5.3.10/15-112	Added <a href="#">Section 15.5.3.10</a> , “Lossless Flow Control Configuration Registers,” and subsections.
15.6.3.8/15-151	Clarified last sentence of the section to include multicast packets.
15.6.4.3/15-160	In <a href="#">Table 15-140</a> , added text to bit field descriptions of IP and PRO.
15.6.6/15-170	Added <a href="#">Section 15.6.6</a> , “Lossless Flow Control,” and subsections.
15.6.7.3/15-178	In <a href="#">Table 15-147</a> , bit field description of “Rx Data Buffer Pointer” has been updated to include a recommendation to use 64-byte aligned receive buffer pointer addresses.
17.1.1.2/17-4	Added a note to the first paragraph, stating that inbound writes less than 4-bytes are split into single bytes writes to the target.
17.2/17-6	In <a href="#">Table 17-2</a> , added PCI_CLK.
18.3.8.1.2/18-44	In <a href="#">Table 18-36</a> , corrected the device ID description.
18.3.10.18/18-94	Clarified that the reset value of the configuration ready register is defined during POR configuration.
19.3.1/19-2	

19.3.2/19-2	In <a href="#">Table 19-1</a> and <a href="#">Table 19-2</a> , changed GPOUT[24:31] to GPOUT[0:7].
19.4/19-3	In <a href="#">Table 19-3</a> , added POR I/O device status register 2 (PORDEVSR2), SerDes 1 control register 1 (SRDS1CR1), and SerDes 2 control register 1 (SRDS2CR1). Also, removed SRDS2CR.
19.4.1/19-4	Added <a href="#">Section 19.4.1.6</a> , “POR Device Status Register 2 (PORDEVSR2),” <a href="#">Section 19.4.1.24</a> , “SerDes 1 Control Register 1 (SRDS1CR1),” and <a href="#">Section 19.4.1.25</a> , “SerDes 2 Control Register 1 (SRDS2CR1).”
20.4.7/20-16	Added <a href="#">Table 20-10</a> . Also added a note about events counted from peripherals on on-chip network being counted twice.
21.3.1.4/21-13	Added <a href="#">Table 21-12</a> .
<a href="#">Appendix B/B-1</a>	In <a href="#">Table B-1</a> , changed “eTSEC2” to “eTSEC3” in the eTSEC register section, as well as in footnote 6.
Index	Added chapter numbers.



# Appendix B

## Complete List of Configuration, Control, and Status Registers

Table B-1. Memory Map

Offset	Register	Access	Reset	Section/Page
<b>Local-Access Registers—Configuration, Control, and Status Registers</b>				
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	<a href="#">4.3.1.1.2/4-5</a>
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	<a href="#">4.3.1.2.1/4-6</a>
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	<a href="#">4.3.1.2.1/4-6</a>
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	<a href="#">4.3.1.3.1/4-7</a>
<b>Local-Access Registers—Local-Access Window Base and Size Registers</b>				
0x0_0BF8	LAIPBRR1—Local access IP block revision register 1	R	0x0000_0000	<a href="#">2.2.3.2/2-6</a>
0x0_0BFC	LAIPBRR2—Local access IP block revision register 2	R	0x0000_0000	<a href="#">2.2.3.3/2-6</a>
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>
0x0_0D08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	<a href="#">2.2.3.4/2-7</a>
0x0_0D10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	<a href="#">2.2.3.5/2-7</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
<b>e500 Coherency Module</b>				
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	<a href="#">8.2.1.1/8-3</a>
0x0_1008	ECM CCB data configuration register (EEBDCR)	4-byte R/W	0x0000_0000	<a href="#">8.2.1.2/8-4</a>
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*00_0000	<a href="#">8.2.1.3/8-5</a>
0x0_1018	ECM CCB debug register (EEBDR)	4-byte R/W		<a href="#">8.2.1.4/8-5</a>
0x0_1040	ECM full threshold count register 0 (EFTCR0)	Mixed	0x8000_0010	<a href="#">8.2.1.5/8-6</a>
0x0_1044	ECM full threshold count register 1 (EFTCR1)	Mixed	0x8000_0010	<a href="#">8.2.1.6/8-7</a>
0x0_1048	ECM full threshold count register 2 (EFTCR2)	Mixed	0x8000_0010	<a href="#">8.2.1.7/8-7</a>
0x0_104C	ECM full threshold count register 3 (EFTCR3)	Mixed	0x8000_0010	<a href="#">8.2.1.8/8-8</a>
0x0_1060	ECM full threshold count register 4 (EFTCR4)	Mixed	0x8000_0010	<a href="#">8.2.1.9/8-9</a>
<b>DDR Memory Controller</b>				
<b>Block Base Address: 0x0_2000</b>				
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	<a href="#">9.4.1.1/9-11</a>
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	<a href="#">9.4.1.2/9-11</a>
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	<a href="#">9.4.1.3/9-13</a>
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	<a href="#">9.4.1.4/9-14</a>
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	<a href="#">9.4.1.5/9-16</a>
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	<a href="#">9.4.1.6/9-18</a>
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	<a href="#">9.4.1.7/9-20</a>
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	<a href="#">9.4.1.8/9-23</a>
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	<a href="#">9.4.1.9/9-25</a>
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	<a href="#">9.4.1.10/9-26</a>
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	<a href="#">9.4.1.11/9-26</a>
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	<a href="#">9.4.1.12/9-29</a>
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	<a href="#">9.4.1.13/9-29</a>
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	<a href="#">9.4.1.14/9-30</a>



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x140–0x147	Reserved	—	—	
0x148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	0x0000_0000	<a href="#">9.4.1.15/9-30</a>
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	<a href="#">9.4.1.16/9-31</a>
0x150–0xBF4	Reserved	—	—	
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0xn <sub>nnn</sub> _n <sub>nnn</sub> <sup>1</sup>	<a href="#">9.4.1.17/9-32</a>
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x0000_0000	<a href="#">9.4.1.18/9-32</a>
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	<a href="#">9.4.1.19/9-33</a>
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	<a href="#">9.4.1.20/9-33</a>
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	<a href="#">9.4.1.21/9-34</a>
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	<a href="#">9.4.1.22/9-34</a>
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	<a href="#">9.4.1.23/9-35</a>
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	<a href="#">9.4.1.24/9-35</a>
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	<a href="#">9.4.1.25/9-35</a>
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	<a href="#">9.4.1.26/9-36</a>
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	<a href="#">9.4.1.27/9-37</a>
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	<a href="#">9.4.1.28/9-38</a>
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	<a href="#">9.4.1.29/9-39</a>
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	<a href="#">9.4.1.30/9-40</a>
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	<a href="#">9.4.1.31/9-40</a>
<b>I<sup>2</sup>C</b>				
<b>I<sup>2</sup>C1 Registers</b>				
<b>Block Base Address: 0x0_3000</b>				
0x000	I2CADR—I <sup>2</sup> C address register	R/W	0x00	<a href="#">11.3.1.1/11-6</a>
0x004	I2CFDR—I <sup>2</sup> C frequency divider register	R/W	0x00	<a href="#">11.3.1.2/11-6</a>
0x008	I2CCR—I <sup>2</sup> C control register	Mixed	0x00	<a href="#">11.3.1.3/11-7</a>
0x00C	I2CSR—I <sup>2</sup> C status register	Mixed	0x81	<a href="#">11.3.1.4/11-9</a>
0x010	I2CDR—I <sup>2</sup> C data register	R/W	0x00	<a href="#">11.3.1.5/11-10</a>
0x014	I2CDFSRR—I <sup>2</sup> C digital filter sampling rate register	R/W	0x10	<a href="#">11.3.1.6/11-11</a>
<b>I<sup>2</sup>C2 Registers</b>				
<b>Block Base Address: 0x0_3000</b>				

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x100– 0x114	I <sup>2</sup> C2 Registers <sup>2</sup>			
<b>DUART Registers</b>				
<b>Block Base Address: 0x0_4000</b>				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	<a href="#">13.3.1.1/13-6</a>
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	<a href="#">13.3.1.2/13-6</a>
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	<a href="#">13.3.1.4/13-9</a>
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	<a href="#">13.3.1.5/13-9</a>
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	<a href="#">13.3.1.6/13-11</a>
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	<a href="#">13.3.1.2/13-6</a>
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	<a href="#">13.3.1.7/13-12</a>
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	<a href="#">13.3.1.8/13-14</a>
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	<a href="#">13.3.1.9/13-15</a>
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	<a href="#">13.3.1.10/13-16</a>
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	<a href="#">13.3.1.11/13-17</a>
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	<a href="#">13.3.1.12/13-17</a>
0x600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	<a href="#">13.3.1.1/13-6</a>
0x600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	<a href="#">13.3.1.2/13-6</a>
0x600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	<a href="#">13.3.1.4/13-9</a>
0x601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	<a href="#">13.3.1.3/13-7</a>
0x602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	<a href="#">13.3.1.5/13-9</a>
0x602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	<a href="#">13.3.1.6/13-11</a>
0x602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	<a href="#">13.3.1.2/13-6</a>
0x603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	<a href="#">13.3.1.7/13-12</a>
0x604	UMCR—ULCR[DLAB] = x UART1 modem control register	R/W	0x00	<a href="#">13.3.1.8/13-14</a>
0x605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	<a href="#">13.3.1.9/13-15</a>
0x606	UMSR—ULCR[DLAB] = x UART1 modem status register	R	0x00	<a href="#">13.3.1.10/13-16</a>
0x607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	<a href="#">13.3.1.11/13-17</a>
<b>Local Bus Controller Registers</b>				
<b>Local bus block base address: 0x0_5000</b>				

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x000	BR0—Base register 0	R/W	0x0000_nn01 <sup>3</sup>	14.3.1.1/14-10
0x008	BR1—Base register 1		0x0000_0000	
0x010	BR2—Base register 2			
0x018	BR3—Base register 3			
0x020	BR4—Base register 4			
0x028	BR5—Base register 5			
0x030	BR6—Base register 6			
0x038	BR7—Base register 7			
0x004	OR0—Options register 0	R/W	0x0000_0FF7	14.3.1.2/14-12
0x00C	OR1—Options register 1		0x0000_0000	
0x014	OR2—Options register 2			
0x01C	OR3—Options register 3			
0x024	OR4—Options register 4			
0x02C	OR5—Options register 5			
0x034	OR6—Options register 6			
0x03C	OR7—Options register 7			
0x068	MAR—UPM address register	R/W	0x0000_0000	14.3.1.3/14-17
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	14.3.1.4/14-17
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	14.3.1.5/14-20
0x088	MDR—UPM data register	R/W	0x0000_0000	14.3.1.6/14-20
0x094	LSDMR—SDRAM mode register	R/W	0x0000_0000	14.3.1.7/14-21
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	14.3.1.8/14-23
0x0A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	14.3.1.9/14-23
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	14.3.1.10/14-24
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	14.3.1.11/14-25
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	14.3.1.12/14-26
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	14.3.1.13/14-27
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	14.3.1.14/14-29
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	14.3.1.15/14-29
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	14.3.1.16/14-30
<b>PCI Controller Memory-Mapped Registers—Block Base Address 0x0_8000</b>				
<b>PCI Configuration Access Registers</b>				
0x000	CFG_ADDR—PCI configuration address	R/W	0x0000_0000	17.3.1.1.1/17-14

**Table B-1. Memory Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x004	CFG_DATA—PCI configuration data	R/W	0x0000_0000	<a href="#">17.3.1.1.2/17-15</a>
0x008	INT_ACK—PCI interrupt acknowledge	R	0x0000_0000	<a href="#">17.3.1.1.3/17-15</a>
0x00C– 0xBFC	Reserved	—	—	—
<b>PCI ATMU Registers—Outbound and Inbound</b>				
0xC00–0xC3C—Outbound Window 0 (default)				
0xC00	POTAR0—PCI outbound window 0 (default) translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC04	POTEAR0—PCI outbound window 0 (default) translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>
0xC08	Reserved	—	—	
0xC0C	Reserved	—	—	
0xC0C	POWAR0—PCI outbound window 0 (default) attributes register	R/W	0x8004_401F	<a href="#">17.3.1.2.4/17-18</a>
0xC14– 0xC1C	Reserved	—	—	
0xC20–0xC3C—Outbound Window 1				
0xC20	POTAR1—PCI outbound window 1 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC24	POTEAR1—PCI outbound window 1 translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>
0xC28	POWBAR1—PCI outbound window 1 base address register	R/W	0x0000_0000	<a href="#">17.3.1.2.3/17-17</a>
0xC2C	Reserved	—	—	
0xC2C	POWAR1—PCI outbound window 1 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.2.4/17-18</a>
0xC30	Reserved	—	—	
0xC40–0xC5C—Outbound Window 2				
0xC40	POTAR2—PCI outbound window 2 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC44	POTEAR2—PCI outbound window 2 translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>
0xC48	POWBAR2—PCI outbound window 2 base address register	R/W	0x0000_0000	<a href="#">17.3.1.2.3/17-17</a>
0xC4C	Reserved	—	—	
0xC50	POWAR2—PCI outbound window 2 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.2.4/17-18</a>
0xC54– 0xC5C	Reserved	—	—	
0xC60–0xC7C—Outbound Window 3				
0xC60	POTAR3—PCI outbound window 3 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC64	POTEAR3—PCI outbound window 3 translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC68	POWBAR3—PCI outbound window 3 base address register	R/W	0x0000_0000	<a href="#">17.3.1.2.3/17-17</a>
0xC6C	Reserved	—	—	
0xC70	POWAR3—PCI outbound window 3 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.2.4/17-18</a>
0xC74– 0xC7C	Reserved	—	—	
0xC80–0xC9C—Outbound Window 4				
0xC80	POTAR4—PCI outbound window 4 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.2.1/17-16</a>
0xC84	POTEAR4—PCI outbound window 4 translation extended address register	R/W	0x0000_0000	<a href="#">17.3.1.2.2/17-16</a>
0xC88	POWBAR4—PCI outbound window 4 base address register	R/W	0x0000_0000	<a href="#">17.3.1.2.3/17-17</a>
0xC8C	Reserved	—	—	
0xC90	POWAR4—PCI outbound window 4 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.2.4/17-18</a>
0xC94– 0xD9C	Reserved	—	—	
0xDA0–0xDBC—Inbound Window 3				
0xDA0	PITAR3—PCI inbound window 3 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0xDA4	Reserved	—	—	
0xDA8	PIWBAR3—PCI inbound window 3 base address register	R/W	0x0000_0000	<a href="#">17.3.1.3.2/17-20</a>
0xDAC	PIWBEAR3—PCI inbound window 3 base extended address register	R/W	0x0000_0000	<a href="#">17.3.1.3.3/17-21</a>
0xDB0	PIWAR3—PCI inbound window 3 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0xDB4– 0xDBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC—Inbound Window 2				
0x0_8DC0	PITAR2—PCI inbound window 2 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI inbound window 2 base address register	R/W	0x0000_0000	
0x0_8DCC	PIWBEAR2—PCI inbound window 2 base extended address register	R/W	0x0000_0000	<a href="#">17.3.1.3.3/17-21</a>
0x0_8DD0	PIWAR2—PCI inbound window 2 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0x0_8DD4– 0x0_8DDC	Reserved	—	—	
0xDE0–0xDFC—Inbound Window 1				
0xDE0	PITAR1—PCI inbound window 1 translation address register	R/W	0x0000_0000	<a href="#">17.3.1.3.1/17-20</a>
0xDE4	Reserved	—	—	
0xDE8	PIWBAR1—PCI inbound window 1 base address register	R/W	0x0000_0000	<a href="#">17.3.1.3.2/17-20</a>
0xDEC	Reserved	—	—	

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xDF0	PIWAR1—PCI inbound window 1 attributes register	R/W	0x0000_0000	<a href="#">17.3.1.3.4/17-21</a>
0xDF4– 0xDFC	Reserved	—	—	
<b>PCI Error Management Registers</b>				
0xE00	ERR_DR—PCI error detect register	w1c	0x0000_0000	<a href="#">17.3.1.4.1/17-24</a>
0xE04	ERR_CAP_DR—PCI error capture disabled register	R/W	0x0000_0000	<a href="#">17.3.1.4.2/17-25</a>
0xE08	ERR_EN—PCI error enable register	R/W	0x0000_0000	<a href="#">17.3.1.4.3/17-26</a>
0xE0C	ERR_ATTRIB—PCI error attributes capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.4/17-27</a>
0xE10	ERR_ADDR—PCI error address capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.5/17-28</a>
0xE14	ERR_EXT_ADDR—PCI error extended address capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.6/17-28</a>
0xE18	ERR_DL—PCI error data low capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.7/17-29</a>
0xE1C	ERR_DH—PCI error data high capture register	R/W	0x0000_0000	<a href="#">17.3.1.4.8/17-29</a>
0xE20	GAS_TIMR—PCI gasket timer register	R/W	0x0100_3FFF	<a href="#">17.3.1.4.9/17-29</a>
0xE24				
0xE28– 0xEFC	Reserved	—	—	
0xF00– 0xFFC	Reserved for debug	—	—	
<b>PCI Express Registers</b>				
<b>PCI Express Controller 1 Memory-Mapped Registers—Block Base Address 0x0_A000</b>				
<b>PCI Express Configuration Access Registers</b>				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	<a href="#">18.3.2.1/18-9</a>
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	<a href="#">18.3.2.2/18-10</a>
0x008	Reserved	—	—	
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	<a href="#">18.3.2.3/18-11</a>
0x010	PEX_CONFIG_RTU_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	<a href="#">18.3.2.4/18-11</a>
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	<a href="#">18.3.2.5/18-12</a>
0x018– 0x01C	Reserved	—	—	
<b>PCI Express Power Management Event &amp; Message Registers</b>				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	<a href="#">18.3.3.1/18-13</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	<a href="#">18.3.3.2/18-14</a>
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	<a href="#">18.3.3.3/18-16</a>
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	<a href="#">18.3.3.4/18-17</a>
0x030–0xBF4	Reserved	—	—	
<b>PCI Express IP Block Revision Registers</b>				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0100	<a href="#">18.3.4.1/18-18</a>
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	<a href="#">18.3.4.2/18-19</a>
<b>PCI Express ATMU Registers</b>				
<b>Outbound Window 0 (default)</b>				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC08–0xC0C	Reserved	—	—	
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC14–0xC1C	Reserved	—	—	
<b>Outbound Window 1</b>				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC2C	Reserved	—	—	
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC34–0xC3C	Reserved	—	—	
<b>Outbound Window 2</b>				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC4C	Reserved	—	—	
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC54– 0xC5C	Reserved	—	—	
<b>Outbound Window 3</b>				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC6C	Reserved	—	—	
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC74– 0xC7C	Reserved	—	—	
<b>Outbound Window 4</b>				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.1/18-20</a>
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.2/18-21</a>
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	<a href="#">18.3.5.1.3/18-21</a>
0xC8C	Reserved	—	—	
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	<a href="#">18.3.5.1.4/18-22</a>
0xC94– 0xC9C	Reserved	—	—	
0xCA0– 0xD9C	Reserved	—	—	
<b>Inbound Window 3</b>				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDA4	Reserved	—	—	
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDAC	PEXIWBAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	<a href="#">18.3.5.2.5/18-27</a>



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>
0xDB4– 0xDBC	Reserved	—	—	
<b>Inbound Window 2</b>				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDC4	Reserved	—	—	
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	<a href="#">18.3.5.2.5/18-27</a>
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>
0xDD4– 0xDDC	Reserved	—	—	
<b>Inbound Window 1</b>				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	<a href="#">18.3.5.2.3/18-25</a>
0xDE4	Reserved	—	—	
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	<a href="#">18.3.5.2.4/18-26</a>
0xDEC	Reserved	—	—	
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	<a href="#">18.3.5.2.6/18-27</a>
0xDF4– 0xDFC	Reserved	—	—	
<b>PCI Express Error Management Registers</b>				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	<a href="#">18.3.6.1/18-29</a>
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	<a href="#">18.3.6.2/18-32</a>
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	<a href="#">18.3.6.3/18-34</a>
0xE14– 0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	<a href="#">18.3.6.4/18-35</a>
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	<a href="#">18.3.6.5/18-36</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	<a href="#">18.3.6.6/18-38</a>
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	<a href="#">18.3.6.7/18-39</a>
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	<a href="#">18.3.6.8/18-40</a>
0xE38– 0xFFC	Reserved	—	—	
<b>PCI Express Controller 2 Memory-Mapped Registers—Block Base Address 0x0_9000</b>				
0x000– 0xFFC	PCI Express Controller 2 registers <b>Note:</b> All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 2; the offsets of PCI Express Controller 2 registers are the same except they have a different block base address.			
<b>PCI Express Controller 3 Memory-Mapped Registers—Block Base Address 0x0_B000</b>				
0x000– 0xFFC	PCI Express Controller 3 registers <b>Note:</b> All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 3; the offsets of PCI Express Controller 3 registers are the same except they have a different block base address.			
<b>L2/SRAM Memory-Mapped Configuration Registers</b>				
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	<a href="#">7.3.1.1/7-10</a>
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	<a href="#">7.3.1.2.1/7-13</a>
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	<a href="#">7.3.1.2.3/7-14</a>
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	<a href="#">7.3.1.2.1/7-13</a>
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	<a href="#">7.3.1.2.3/7-14</a>
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	<a href="#">7.3.1.2.1/7-13</a>
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	<a href="#">7.3.1.2.3/7-14</a>
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	<a href="#">7.3.1.2.1/7-13</a>
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	<a href="#">7.3.1.2.3/7-14</a>
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	<a href="#">7.3.1.3.1/7-16</a>
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	<a href="#">7.3.1.3.1/7-16</a>
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	<a href="#">7.3.1.4.1/7-18</a>
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	<a href="#">7.3.1.4.1/7-18</a>
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	<a href="#">7.3.1.4.1/7-18</a>
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E40	L2ERRDET—L2 error detect register	w1c	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E50	L2ERRADDRH—L2 error address capture register high	R	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E54	L2ERRADDRL—L2 error address capture register low	R	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	<a href="#">7.3.1.4.2/7-20</a>
<b>DMA Registers</b>				
<b>DMA Controller Block Base Address: 0x2_1000</b>				
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	<a href="#">16.3.1.1/16-10</a>
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	<a href="#">16.3.1.2/16-12</a>
0x108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	<a href="#">16.3.1.4/16-15</a>
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	<a href="#">16.3.1.5/16-16</a>
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	<a href="#">16.3.1.6/16-17</a>
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	<a href="#">16.3.1.7/16-18</a>
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	<a href="#">16.3.1.8/16-19</a>
0x124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.9/16-19</a>
0x130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.10/16-21</a>
0x138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	<a href="#">16.3.1.11/16-22</a>
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	<a href="#">16.3.1.12/16-22</a>
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	<a href="#">16.3.1.13/16-23</a>
0x148– 0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	<a href="#">16.3.1.1/16-10</a>
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	<a href="#">16.3.1.2/16-12</a>
0x188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	<a href="#">16.3.1.3/16-13</a>
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	<a href="#">16.3.1.4/16-15</a>
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	<a href="#">16.3.1.5/16-16</a>
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	<a href="#">16.3.1.6/16-17</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	16.3.1.7/16-18
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	16.3.1.8/16-19
0x1A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	16.3.1.9/16-19
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	16.3.1.9/16-19
0x1B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	16.3.1.10/16-21
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	16.3.1.10/16-21
0x1B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	16.3.1.11/16-22
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	16.3.1.11/16-22
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	16.3.1.12/16-22
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	16.3.1.13/16-23
0x1C8– 0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	16.3.1.1/16-10
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	16.3.1.2/16-12
0x208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	16.3.1.3/16-13
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	16.3.1.3/16-13
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	16.3.1.4/16-15
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	16.3.1.5/16-16
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	16.3.1.6/16-17
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	16.3.1.7/16-18
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	16.3.1.8/16-19
0x224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	16.3.1.9/16-19
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	16.3.1.9/16-19
0x230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	16.3.1.10/16-21
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	16.3.1.10/16-21
0x238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	16.3.1.11/16-22
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	16.3.1.11/16-22
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	16.3.1.12/16-22
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	16.3.1.13/16-23
0x248– 0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	16.3.1.1/16-10

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	16.3.1.2/16-12
0x288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	16.3.1.3/16-13
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	16.3.1.3/16-13
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	16.3.1.4/16-15
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	16.3.1.5/16-16
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	16.3.1.6/16-17
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	16.3.1.7/16-18
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	16.3.1.8/16-19
0x2A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	16.3.1.9/16-19
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	16.3.1.9/16-19
0x2B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	16.3.1.10/16-21
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	16.3.1.10/16-21
0x2B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	16.3.1.11/16-22
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	16.3.1.11/16-22
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	16.3.1.12/16-22
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	16.3.1.13/16-23
0x2C8– 0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	16.3.1.14/16-24
<b>eTSEC Registers<sup>4</sup></b>				
<b>eTSEC<sub>n</sub> General Control and Status Registers</b>				
0x2_4000	TSEC_ID*—Controller ID register	R	0x0124_0000	15.5.3.1.1/15-22
0x2_4004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	15.5.3.1.2/15-23
0x2_4008– 0x2_400C	Reserved	R	0x0030_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	15.5.3.1.3/15-24
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	15.5.3.1.4/15-28
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	15.5.3.1.5/15-30
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	15.5.3.1.6/15-31
0x2_4024	Reserved	R	0x0000_0000	—
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	15.5.3.1.7/15-33
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	15.5.3.1.8/15-34

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	15.5.3.1.9/15-35
0x2_4034– 0x2_404C	Reserved	R	0x0000_0000	
<b>eTSEC Transmit Control and Status Registers</b>				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	15.5.3.2.1/15-36
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	15.5.3.2.2/15-38
0x2_4108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	15.5.3.2.3/15-42
0x2_410C	Reserved	R	0x0000_0000	—
0x2_4110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	15.5.3.2.4/15-43
0x2_4114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	15.5.3.2.5/15-44
0x2_4118– 0x2_413C	Reserved	R	0x0000_0000	—
0x2_4140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	15.5.3.2.6/15-44
0x2_4144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	15.5.3.2.7/15-45
0x2_4148– 0x2_417C	Reserved	R	0x0000_0000	—
0x2_4180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_4184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_4188	Reserved	R	0x0000_0000	—
0x2_418C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_4190	Reserved	R	0x0000_0000	—
0x2_4194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_4198	Reserved	R	0x0000_0000	—
0x2_419C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_41A0	Reserved	R	0x0000_0000	—
0x2_41A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_41A8	Reserved	R	0x0000_0000	—
0x2_41AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_41B0	Reserved	R	0x0000_0000	—
0x2_41B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_41B8	Reserved	R	0x0000_0000	—
0x2_41BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.2.9/15-46
0x2_41C0– 0x2_41FC	Reserved	R	0x0000_0000	—
0x2_4200	TBASEH—TxBD base address high bits	R/W	0x0000_0000	15.5.3.2.10/15-47
0x2_4204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	15.5.3.2.11/15-48
0x2_4208	Reserved	R	0x0000_0000	—

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_420C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4210	Reserved	R	0x0000_0000	
0x2_4214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4218	Reserved	R	0x0000_0000	—
0x2_421C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4220	Reserved	R	0x0000_0000	—
0x2_4224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4228	Reserved	R	0x0000_0000	—
0x2_422C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4230	Reserved	R	0x0000_0000	—
0x2_4234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4238	Reserved	R	0x0000_0000	—
0x2_423C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	<a href="#">15.5.3.2.11/15-48</a>
0x2_4240– 0x2_42FC	Reserved	R	0x0000_0000	—
<b>eTSEC Receive Control and Status Registers</b>				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	<a href="#">15.5.3.3.1/15-48</a>
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	<a href="#">15.5.3.3.2/15-50</a>
0x2_4308– 0x2_430C	Reserved	R	0x0000_0000	—
0x2_4310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	<a href="#">15.5.3.3.3/15-52</a>
0x2_4314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	<a href="#">15.5.3.3.4/15-53</a>
0x2_4318– 0x2_432C	Reserved	R	0x0000_0000	—
0x2_4330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	<a href="#">15.5.3.3.5/15-55</a>
0x2_4334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	<a href="#">15.5.3.3.6/15-56</a>
0x2_4338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	<a href="#">15.5.3.3.7/15-57</a>
0x2_433C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	<a href="#">15.5.3.3.8/15-58</a>
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	<a href="#">15.5.3.3.9/15-61</a>
0x2_4344– 0x2_437C	Reserved	R	0x0000_0000	—
0x2_4380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_4384	RPTR0—RxBd pointer for ring 0	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_4388	Reserved	R	0x0000_0000	—
0x2_438C	RPTR1*—RxBd pointer for ring 1	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_4390	Reserved	R	0x0000_0000	—
0x2_4394	RPTR2*—RxBd pointer for ring 2	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4398	Reserved	R	0x0000_0000	—
0x2_439C	RBPTR3*—RxBd pointer for ring 3	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_43A0	Reserved	R	0x0000_0000	—
0x2_43A4	RBPTR4*—RxBd pointer for ring 4	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_43A8	Reserved	R	0x0000_0000	—
0x2_43AC	RBPTR5*—RxBd pointer for ring 5	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_43B0	Reserved	R	0x0000_0000	—
0x2_43B4	RBPTR6*—RxBd pointer for ring 6	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_43B8	Reserved	R	0x0000_0000	—
0x2_43BC	RBPTR7*—RxBd pointer for ring 7	R/W	0x0000_0000	<a href="#">15.5.3.3.11/15-62</a>
0x2_43C0– 0x2_43FC	Reserved	R/W	0x0000_0000	—
0x2_4400	RBASEH—RxBd base address high bits	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4404	RBASE0—RxBd base address of ring 0	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4408	Reserved	R	0x0000_0000	—
0x2_440C	RBASE1*—RxBd base address of ring 1	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4410	Reserved	R	0x0000_0000	—
0x2_4414	RBASE2*—RxBd base address of ring 2	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4418	Reserved	R	0x0000_0000	—
0x2_441C	RBASE3*—RxBd base address of ring 3	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4420	Reserved	R	0x0000_0000	—
0x2_4424	RBASE4*—RxBd base address of ring 4	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4428	Reserved	R	0x0000_0000	—
0x2_442C	RBASE5*—RxBd base address of ring 5	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4430	Reserved	R	0x0000_0000	—
0x2_4434	RBASE6*—RxBd base address of ring 6	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4438	Reserved	R	0x0000_0000	—
0x2_443C	RBASE7*—RxBd base address of ring 7	R/W	0x0000_0000	<a href="#">15.5.3.3.13/15-63</a>
0x2_4440– 0x2_44FC	Reserved	R/W	0x0000_0000	—
<b>eTSEC MAC Registers</b>				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	<a href="#">15.5.3.5.1/15-66</a>
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	<a href="#">15.5.3.5.2/15-68</a>
0x2_4508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	<a href="#">15.5.3.5.3/15-70</a>
0x2_450C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	<a href="#">15.5.3.5.4/15-71</a>
0x2_4510	MAXFRM—Maximum frame size	R/W	0x0000_0600	<a href="#">15.5.3.5.5/15-72</a>



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4514– 0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration	R/W	0x0000_0007	<a href="#">15.5.3.5.6/15-72</a>
0x2_4524	MIIMCOM—MII management command	R/W	0x0000_0000	<a href="#">15.5.3.5.7/15-73</a>
0x2_4528	MIIMADD—MII management address	R/W	0x0000_0000	<a href="#">15.5.3.5.8/15-74</a>
0x2_452C	MIIMCON—MII management control	WO	0x0000_0000	<a href="#">15.5.3.5.9/15-74</a>
0x2_4530	MIIMSTAT—MII management status	R	0x0000_0000	<a href="#">15.5.3.5.10/15-75</a>
0x2_4534	MIIMIND—MII management indicator	R	0x0000_0000	<a href="#">15.5.3.5.11/15-75</a>
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status	R	0x0000_0000	<a href="#">15.5.3.5.12/15-76</a>
0x2_4540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	<a href="#">15.5.3.5.13/15-76</a>
0x2_4544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	<a href="#">15.5.3.5.15/15-78</a>

**Table B-1. Memory Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x2_4548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	<a href="#">15.5.3.5.15/15-78</a> <a href="#">15.5.3.5.16/15-78</a>
0x2_454C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x2_4550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x2_4554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x2_4558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x2_455C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x2_4560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x2_4564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x2_4568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x2_456C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x2_4570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x2_4574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x2_4578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x2_457C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x2_4580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x2_4584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x2_4588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x2_458C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x2_4590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	
0x2_4594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	
0x2_4598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	
0x2_459C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x2_45A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x2_45A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x2_45A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x2_45AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x2_45B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x2_45B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x2_45B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x2_45BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
0x2_45C0– 0x2_467C	Reserved	R	0x0000_0000	—
<b>eTSEC Transmit and Receive Counters</b>				
0x2_4680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.1/15-79</a>
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.2/15-80</a>
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.3/15-80</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	15.5.3.6.4/15-81
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	15.5.3.6.5/15-81
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	15.5.3.6.6/15-82
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	15.5.3.6.7/15-82
<b>eTSEC Receive Counters</b>				
0x2_469C	RBYT—Receive byte counter	R/W	0x0000_0000	15.5.3.6.8/15-83
0x2_46A0	RPKT—Receive packet counter	R/W	0x0000_0000	15.5.3.6.9/15-83
0x2_46A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	15.5.3.6.10/15-84
0x2_46A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	15.5.3.6.11/15-84
0x2_46AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	15.5.3.6.12/15-85
0x2_46B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	15.5.3.6.13/15-85
0x2_46B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	15.5.3.6.14/15-86
0x2_46B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	15.5.3.6.15/15-86
0x2_46BC	RALN—Receive alignment error counter	R/W	0x0000_0000	15.5.3.6.16/15-87
0x2_46C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	15.5.3.6.17/15-87
0x2_46C4	RCDE—Receive code error counter	R/W	0x0000_0000	15.5.3.6.18/15-88
0x2_46C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	15.5.3.6.19/15-88
0x2_46CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	15.5.3.6.20/15-89
0x2_46D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	15.5.3.6.21/15-89
0x2_46D4	RFRG—Receive fragments counter	R/W	0x0000_0000	15.5.3.6.22/15-90
0x2_46D8	RJBR—Receive jabber counter	R/W	0x0000_0000	15.5.3.6.23/15-90
0x2_46DC	RDRP—Receive drop counter	R/W	0x0000_0000	15.5.3.6.24/15-91
<b>eTSEC Transmit Counters</b>				
0x2_46E0	TBYT—Transmit byte counter	R/W	0x0000_0000	15.5.3.6.25/15-91
0x2_46E4	TPKT—Transmit packet counter	R/W	0x0000_0000	15.5.3.6.26/15-92
0x2_46E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	15.5.3.6.27/15-92
0x2_46EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	15.5.3.6.28/15-93
0x2_46F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	15.5.3.6.29/15-93
0x2_46F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	15.5.3.6.30/15-94
0x2_46F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	15.5.3.6.31/15-94
0x2_46FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	15.5.3.6.32/15-95
0x2_4700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	15.5.3.6.33/15-95
0x2_4704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	15.5.3.6.34/15-96

**Table B-1. Memory Map (continued)**

Offset	Register	Access	Reset	Section/Page
0x2_4708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.35/15-96</a>
0x2_470C	TNCL—Transmit total collision counter	R/W	0x0000_0000	<a href="#">15.5.3.6.36/15-97</a>
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.37/15-97</a>
0x2_4718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.38/15-98</a>
0x2_471C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	<a href="#">15.5.3.6.39/15-98</a>
0x2_4720	TXCF—Transmit control frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.40/15-99</a>
0x2_4724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.41/15-99</a>
0x2_4728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.42/15-100</a>
0x2_472C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	<a href="#">15.5.3.6.43/15-100</a>
<b>eTSEC Counter Control and TOE Statistics Registers</b>				
0x2_4730	CAR1—Carry register one register <sup>5</sup>	R	0x0000_0000	<a href="#">15.5.3.6.44/15-100</a>
0x2_4734	CAR2—Carry register two register <sup>6</sup>	R	0x0000_0000	<a href="#">15.5.3.6.45/15-103</a>
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	<a href="#">15.5.3.6.46/15-104</a>
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	<a href="#">15.5.3.6.47/15-105</a>
0x2_4740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	<a href="#">15.5.3.6.48/15-106</a>
0x2_4744– 0x2_47FC	Reserved	R	0x0000_0000	—
<b>Hash Function Registers</b>				
0x2_4800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	<a href="#">15.5.3.7.1/15-107</a>
0x2_4804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x2_4808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x2_480C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x2_4810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x2_4814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x2_4818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x2_481C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	
0x2_4820– 0x2_487C	Reserved	R	0x0000_0000	—

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	15.5.3.7.2/15-108
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0– 0x2_49FC	Reserved	R	0x0000_0000	—
<b>eTSEC FIFO Control Registers</b>				
0x2_4A00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_00C0	15.5.3.8.1/15-109
0x2_4A04– 0x2_4AFC	Reserved	R	0x0000_0000	—
<b>eTSEC<sub>n</sub> DMA Attribute Registers</b>				
0x2_4B00– 0x2_4BF4	Reserved	R	0x0000_0000	—
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	15.5.3.9.1/15-110
0x2_4BFC	ATTRELI—Attribute extract length and extract index register	R/W	0x0000_0000	15.5.3.9.2/15-111
<b>eTSEC Future Expansion Space</b>				
0x2_4C00– 0x2_4D94	Reserved	—	—	—
0x2_4D98– 0x2_4FFF	Reserved	R	0x0000_0000	—
0x2_6000– 0x2_6FFF	eTSEC3 REGISTERS <sup>6</sup>			
<b>Integrated Security Engine</b>				
<b>Controller</b>				
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	12.6.5.2/12-110
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	12.6.5.3/12-111
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	12.6.5.4/12-112
0x3_1020	ID—Identification register	R	0x0000_0000_0000_0040	12.6.5.5/12-113
0x3_1BF8	IP block revision	R	0x0030_0000_0000_0000	12.6.5.6/12-113

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	12.6.5.1/12-109
0x3_1030	MCR—Master control register	R/W	0000_0000_00_00_0000	12.6.5.7/12-114
<b>Channel 1</b>				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000_0000_0000	12.5.1.1/12-92
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	12.5.1.2/12-95
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000_0000_0000	12.5.1.3/12-100
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000_0000_0000	12.5.1.4/12-101
0x3_1180–0x3_11BF	DB1—Crypto-channel 1 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	12.5.1.5/12-102
0x3_11C0–0x3_11DF	Gather link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
0x3_11E0–0x3_11FF	Scatter link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
<b>Channel 2</b>				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000_0000_0000	12.5.1.1/12-92
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	12.5.1.2/12-95
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000_0000_0000	12.5.1.3/12-100
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000_0000_0000	12.5.1.4/12-101
0x3_1280–0x3_12BF	DB2—Crypto-channel 2 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	12.5.1.5/12-102
0x3_12C0–0x3_12DF	Gather link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
0x3_12E0–0x3_12FF	Scatter link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
<b>Channel 3</b>				
0x3_1308	CCCR3—Crypto-channel3 configuration register	R/W	0x0000_0000_0000_0000	12.5.1.1/12-92
0x3_1310	CCPSR3—Crypto-channel3 pointer status register	R	0x0000_0000_0000_0007	12.5.1.2/12-95
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000_0000_0000	12.5.1.3/12-100

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000_0000_0000	12.5.1.4/12-101
0x3_1380–0x3_13BF	DB3—Crypto-channel 3 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	12.5.1.5/12-102
0x3_13C0–0x3_13DF	Gather link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
0x3_13E0–0x3_13FF	Scatter link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
<b>Channel 4</b>				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000_0000_0007	12.5.1.1/12-92
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0000	12.5.1.2/12-95
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000_0000_0000	12.5.1.3/12-100
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000_0000_0000	12.5.1.4/12-101
0x3_1480–0x3_14BF	DB4—Crypto-channel 4 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	12.5.1.5/12-102
0x3_14C0–0x3_14DF	Gather link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
0x3_14E0–0x3_14FF	Scatter link tables	W	0x0000_0000_0000_0000	12.5.1.6/12-102
<b>Data Encryption Standard Execution Unit (DEU)</b>				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	12.4.2.1/12-33
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	12.4.2.2/12-34
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000_0000_0000	12.4.2.3/12-35
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000_0000_0000	12.4.2.4/12-35
0x3_2028	DEUSR—DEU status register	R	0x0000_0000_0000_0000	12.4.2.5/12-36
0x3_2030	DEUI SR—DEU interrupt status register	R	0x0000_0000_0000_0000	12.4.2.6/12-37
0x3_2038	DEUI CR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	12.4.2.7/12-39
0x3_2050	DEUEUG—DEU EU go register	W	0x0000_0000_0000_0000	12.4.2.8/12-40
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000_0000_0000	12.4.2.9/12-41

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_2400	DEUK1—DEU key register 1	W	—	12.4.2.10/12-41
0x3_2408	DEUK2—DEU key register 2	W	—	12.4.2.10/12-41
0x3_2410	DEUK3—DEU key register 3	W	—	12.4.2.10/12-41
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000_0000_0000	12.4.2.11/12-41
<b>Advanced Encryption Standard Execution Unit (AESU)</b>				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000_0000_0000	12.4.6.1/12-67
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000_0000_0000	12.4.6.2/12-69
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000_0000_0000	12.4.6.3/12-69
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000_0000_0000	12.4.6.4/12-70
0x3_4028	AESUSR—AESU status register	R	0x0000_0000_0000_0000	12.4.6.5/12-71
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000_0000_0000	12.4.6.6/12-72
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000_0000_1000	12.4.6.7/12-73
0x3_4050	AESUEUG—AESU EU go register	W	0x0000_0000_0000_0000	12.4.6.8/12-74
0x3_4100–0x3_4108	AESU context memory registers	R/W	0x0000_0000_0000_0000	12.4.6.9/12-75
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000_0000_0000	12.4.6.9/12-75
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000_0000_0000	12.4.6.9/12-75
<b>Message Digest Execution Unit (MDEU)</b>				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000_0000_0000	12.4.4.1/12-50
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000_0000_0000	12.4.4.2/12-53
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000_0000_0000	12.4.4.2/12-53
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000_0000_0000	12.4.4.5/12-55
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000_0000_0000	12.4.4.6/12-55
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000_0000_0000	12.4.4.7/12-57



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000_0000_1000	12.4.4.8/12-58
0x3_6040	MDEUICVSR—MDEU ICV size register	W	0x0000_0000_0000_0000	12.4.4.9/12-59
0x3_6050	MDEUEUG—MDEU EU go register	W	0x0000_0000_0000_0000	12.4.4.10/12-59
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000_0000_0000	12.4.4.11/12-60
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000_0000_0000	12.4.4.12/12-61
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000_0000_0000	12.4.4.13/12-62
<b>ARC Four Execution Unit (AFEU)</b>				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000_0000_0000	12.4.3.1/12-42
0x3_808	AFEUKSR—AFEU key size register	R/W	0x0000_0000_0000_0000	12.4.3.3/12-43
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000_0000_0000	12.4.3.4/12-44
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000_0000_0000	12.4.3.5/12-44
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000_0000_0000	12.4.3.6/12-45
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000_0000_0000	12.4.3.7/12-46
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	12.4.3.8/12-48
0x3_8050	AFEUEUG—AFEU EU go register	W	0x0000_0000_0000_0000	12.4.3.9/12-49
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000_0000_0000	12.4.3.10.1/12-49
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000_0000_0000	12.4.3.10.2/12-50
0x3_8400	AFEUK1—AFEU key register 0	W	—	12.4.3.11/12-50
0x3_8480	AFEUK2—AFEU key register 1	W	—	12.4.3.11/12-50
0x3_8800–0x3_8FFF (3_8E00)	AFEU FIFO	R/W	0x0000_0000_0000_0000	12.4.3.12/12-50
<b>Random Number Generator (RNG)</b>				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000_0000_0000	12.4.5.1/12-63

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000_0000_0000	12.4.5.2/12-63
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000_0000_0000	12.4.5.3/12-63
0x3_A028	RNGSR—RNG status register	R	0x0000_0000_0000_0000	12.4.5.4/12-64
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000_0000_0000	12.4.5.5/12-65
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000_0000_1000	12.4.5.6/12-66
0x3_A050	RNGEUG—RNG EU go register	W	0x0000_0000_0000_0000	12.4.5.7/12-67
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000_0000_0000	12.4.5.8/12-67
<b>Public Key Execution Units (PKEU)</b>				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000_0000_0000	12.4.1.1/12-26
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000_0000_0000	12.4.1.2/12-27
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000_0000_0000	12.4.1.4/12-28
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000_0000_0000	12.4.1.5/12-29
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000_0000_0000	12.4.1.6/12-29
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000_0000_0000	12.4.1.7/12-30
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000_0000_1000	12.4.1.8/12-31
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000_0000_0000	12.4.1.3/12-28
0x3_C050	PKEUEUG—PKEU EU go register	W	0x0000_0000_0000_0000	12.4.1.9/12-32
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	12.4.1.10/12-32
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	12.4.1.10/12-32
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	12.4.1.10/12-32
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	12.4.1.10/12-32
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	12.4.1.10/12-32

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C340– 0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_ 0000_0000	12.4.1.10/12-32
0x3_C380– 0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_ 0000_0000	12.4.1.10/12-32
0x3_C3C0– 0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_ 0000_0000	12.4.1.10/12-32
0x3_C400– 0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_ 0000_0000	12.4.1.10/12-32
0x3_C800– 0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_ 0000_0000	12.4.1.10/12-32
<b>Kasumi Execution Unit (KEU)</b>				
0x3_E000	KEUMR—KEU mode register	R/W	0x0000_0000_ 0000_0000	12.4.7.1/12-80
0x3_E008	KEUKSR—KEU key size register	R/W	0x0000_0000_ 0000_0000	12.4.7.2/12-81
0x3_E010	KEUDSR—KEU data size register	R/W	0x0000_0000_ 0000_0000	12.4.7.3/12-81
0x3_E018	KEURCR—KEU reset control register	R/W	0x0000_0000_ 0000_0000	12.4.7.4/12-82
0x3_E028	KEUSR—KEU status register	R	0x0000_0000_ 0000_0000	12.4.7.5/12-83
0x3_E030	KEUISR—KEU interrupt status register	R/W	0x0000_0000_ 0000_0000	12.4.7.6/12-84
0x3_E038	KEUICR—KEU interrupt control register	R/W	0x0000_0000_ 0000_1000	12.4.7.7/12-85
0x3_E048	KEUDOR—KEU data out register (F9 MAC)	R	0x0000_0000_ 0000_0000	12.4.7.8/12-87
0x3_E050	KEUEUG—KEU EU go register	W	0x0000_0000_ 0000_0000	12.4.7.9/12-87
0x3_E100	KEUIV1—KEU initialization vector 1 register	R/W	0x0000_0000_ 0000_0000	12.4.7.10/12-88
0x3_E108	KEUICV—KEU ICV_In register	R/W	0x0000_0000_ 0000_0000	12.4.7.11/12-88
0x3_E110	KEUIV2—KEU initialization vector 2 register (Fresh)	R/W	0x0000_0000_ 0000_0000	12.4.7.12/12-89
0x3_E118	KEUC1—KEU context_1 register	R/W	0x0000_0000_ 0000_0000	12.4.7.13/12-89
0x3_E120	KEUC2—KEU context_2 register	R/W	0x0000_0000_ 0000_0000	12.4.7.14/12-89
0x3_E128	KEUC3—KEU context_3 register	R/W	0x0000_0000_ 0000_0000	12.4.7.15/12-90
0x3_E130	KEUC4—KEU context_4 register	R/W	0x0000_0000_ 0000_0000	12.4.7.13/12-89

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_E138	KEUC5—KEU context_5 register	R/W	0x0000_0000_0000_0000	12.4.7.13/12-89
0x3_E140	KEUC6—KEU context_6 register	R/W	0x0000_0000_0000_0000	12.4.7.13/12-89
0x3_E400	KEUKD1—KEU key data register_1 (CK-high)	R/W	0x0000_0000_0000_0000	12.4.7.14/12-89
0x3_E408	KEUKD2—KEU key data register_2 (CK-low)	R/W	0x0000_0000_0000_0000	12.4.7.14/12-89
0x3_E410	KEUKD3—KEU key data register_3 (IK-high)	R/W	0x0000_0000_0000_0000	12.4.7.15/12-90
0x3_E418	KEUKD4—KEU key data register_4 (IK-low)	R/W	0x0000_0000_0000_0000	12.4.7.15/12-90
0x3_E800–0x3_EFFF	KEUFIFO	R/W	0x0000_0000_0000_0000	12.4.7.16/12-90
<b>Programmable Interrupt Controller Register Address Map</b>				
<b>Global Registers</b>				
0x4_0000	BRR1—Block Revision register 1	R	0x0040_0200	10.3.1.1/10-18
0x4_0010	BRR2—Block Revision register 2	R	0x0000_0001	10.3.1.2/10-19
0x4_0020–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.8.1/10-45
0x4_0050	IPIDR1—Interprocessor interrupt 1 (IPI 1) dispatch register			
0x4_0060	IPIDR2—Interprocessor interrupt 2 (IPI 2) dispatch register			
0x4_0070	IPIDR3—Interprocessor interrupt 3 (IPI 3) dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.8.2/10-46
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.8.3/10-47
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.8.4/10-47
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.8.5/10-48
0x4_00C0–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x004F_0002	10.3.1.3/10-19
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.4/10-20
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.5/10-21
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.6/10-21

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.7/10-22
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.8/10-23
<b>Global Timer Registers</b>				
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-23
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-24
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-24
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-25
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-26
0x4_1200– 0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-26
<b>External, <math>\overline{\text{IRQ\_OUT}}</math>, and Critical Interrupt Summary Registers</b>				
0x4_1308	ERQSR—External interrupt summary register	R	0x0000_0000	10.3.3.1/10-28
0x4_1310	IRQSR0— $\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.2/10-29
0x4_1320	IRQSR1— $\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.3/10-30
0x4_1324	IRQSR2— $\overline{\text{IRQ\_OUT}}$ summary register 2	R	0x0000_0000	10.3.3.4/10-30
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.5/10-31
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.6/10-31
0x4_1344	CISR2—Critical Interrupt summary register 2	R	0x0000_0000	10.3.3.7/10-32
<b>Performance Monitor Mask Registers</b>				

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-32
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-33
0x4_13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_FFFF	10.3.4.3/10-34
0x4_13D0– 0x4_13F0	Reserved	—	—	—
<b>Message Registers</b>				
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-34
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-35
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-35
0x4_1520– 0x4_FFF0	Reserved	—	—	—
0x4_1600	MSIR0—Message shared interrupt register 0	Special	0x0000_0000	10.3.6.1/10-36
0x4_1610	MSIR1—Message shared interrupt register 1	Special	0x0000_0000	10.3.6.1/10-36
0x4_1620	MSIR2—Message shared interrupt register 2	Special	0x0000_0000	10.3.6.1/10-36
0x4_1630	MSIR3—Message shared interrupt register 3	Special	0x0000_0000	10.3.6.1/10-36
0x4_1640	MSIR4—Message shared interrupt register 4	Special	0x0000_0000	10.3.6.1/10-36
0x4_1650	MSIR5—Message shared interrupt register 5	Special	0x0000_0000	10.3.6.1/10-36
0x4_1660	MSIR6—Message shared interrupt register 6	Special	0x0000_0000	10.3.6.1/10-36
0x4_1670	MSIR7—Message shared interrupt register 7	Special	0x0000_0000	10.3.6.1/10-36
0x4_1680– 0x4_1700	Reserved	—	—	—
0x4_1720	MSISR—Message shared interrupt status register	R	0x0000_0000	10.3.6.2/10-36
0x4_1740	MSIIR—Message shared interrupt index register	W	0x0000_0000	10.3.6.3/10-37

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1750– 0x4_FFFF	Reserved	—	—	
<b>PIC Register Address Map—Interrupt Source Configuration Registers</b>				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.1/10-39</a>
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.2/10-40</a>
0x5_0180– 0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0210	IIDR0—Internal interrupt 0 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0230	IIDR1—Internal interrupt 1 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0250	IIDR2—Internal interrupt 2 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0270	IIDR3—Internal interrupt 3 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0290	IIDR4—Internal interrupt 4 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02B0	IIDR5—Internal interrupt 5 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02D0	IIDR6—Internal interrupt 6 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_02F0	IIDR7—Internal interrupt 7 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0310	IIDR8—Internal interrupt 8 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0330	IIDR9—Internal interrupt 9 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0350	IIDR10—Internal interrupt 10 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0370	IIDR11—Internal interrupt 11 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0390	IIDR12—Internal interrupt 12 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03B0	IIDR13—Internal interrupt 13 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03D0	IIDR14—Internal interrupt 14 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_03F0	IIDR15—Internal interrupt 15 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0410	IIDR16—Internal interrupt 16 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0430	IIDR17—Internal interrupt 17 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0450	IIDR18—Internal interrupt 18 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0470	IIDR19—Internal interrupt 19 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0490	IIDR20—Internal interrupt 20 destination register	Mixed	0x0000_0001	10.3.7.4/10-42



Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04B0	IIDR21—Internal interrupt 21 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04D0	IIDR22—Internal interrupt 22 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_04F0	IIDR23—Internal interrupt 23 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0510	IIDR24—Internal interrupt 24 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0530	IIDR25—Internal interrupt 25 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0550	IIDR26—Internal interrupt 26 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0570	IIDR27—Internal interrupt 27 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0590	IIDR28—Internal interrupt 28 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05B0	IIDR29—Internal interrupt 29 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05D0	IIDR30—Internal interrupt 30 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_05F0	IIDR31—Internal interrupt 31 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0600	IIVPR32—Internal interrupt 32 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0610	IIDR32—Internal interrupt 32 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0620	IIVPR33—Internal interrupt 33 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0630	IIDR33—Internal interrupt 33 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0640	IIVPR34—Internal interrupt 34 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0650	IIDR34—Internal interrupt 34 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0660	IIVPR35—Internal interrupt 35 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0670	IIDR35—Internal interrupt 35 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_0680	IIVPR36—Internal interrupt 36 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_0690	IIDR36—Internal interrupt 36 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_06A0	IIVPR37—Internal interrupt 37 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_06B0	IIDR37—Internal interrupt 37 destination register	Mixed	0x0000_0001	10.3.7.4/10-42
0x5_06C0	IIVPR38—Internal interrupt 38 vector/priority register	Mixed	0x8080_0000	10.3.7.3/10-41
0x5_06D0	IIDR38—Internal interrupt 38 destination register	Mixed	0x0000_0001	10.3.7.4/10-42

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_06E0	IIVPR39—Internal interrupt 39 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_06F0	IIDR39—Internal interrupt 39 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0700	IIVPR40—Internal interrupt 40 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0710	IIDR40—Internal interrupt 40 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0720	IIVPR41—Internal interrupt 41 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0730	IIDR41—Internal interrupt 41 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0740	IIVPR42—Internal interrupt 42 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0750	IIDR42—Internal interrupt 42 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0760	IIVPR43—Internal interrupt 43 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0770	IIDR43—Internal interrupt 43 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0780	IIVPR44—Internal interrupt 44 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_0790	IIDR44—Internal interrupt 44 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_07A0	IIVPR45—Internal interrupt 45 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_07B0	IIDR45—Internal interrupt 45 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_07C0	IIVPR46—Internal interrupt 46 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_07D0	IIDR46—Internal interrupt 46 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_07E0	IIVPR47—Internal interrupt 47 vector/priority register	Mixed	0x8080_0000	<a href="#">10.3.7.3/10-41</a>
0x5_07F0	IIDR47—Internal interrupt 47 destination register	Mixed	0x0000_0001	<a href="#">10.3.7.4/10-42</a>
0x5_0800– 0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.5/10-43</a>
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.6/10-43</a>
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.5/10-43</a>
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.6/10-43</a>
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.5/10-43</a>
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.6/10-43</a>
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	Mixed	0x8000_0000	<a href="#">10.3.7.5/10-43</a>
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	Mixed	0x0000_0001	<a href="#">10.3.7.6/10-43</a>
0x5_1680– 0x5_1BF0	Reserved	—	—	—
0x5_1C00	MSIVPR0—Message shared interrupt vector/priority register 0	Mixed	0x1000_0000	<a href="#">10.3.6.4/10-38</a>
0x5_1C10	MSIDR0—Message shared interrupt destination register 0	Mixed	0x0000_0001	<a href="#">10.3.6.5/10-38</a>

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_1C20	MSIVPR1—Message shared interrupt vector/priority register 1	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C30	MSIDR1—Message shared interrupt destination register 1	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C40	MSIVPR2—Message shared interrupt vector/priority register 2	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C50	MSIDR2—Message shared interrupt destination register 2	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C60	MSIVPR3—Message shared interrupt vector/priority register 3	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C70	MSIDR3—Message shared interrupt destination register 3	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1C80	MSIVPR4—Message shared interrupt vector/priority register 4	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1C90	MSIDR4—Message shared interrupt destination register 4	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CA0	MSIVPR5—Message shared interrupt vector/priority register 5	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1CB0	MSIDR5—Message shared interrupt destination register 5	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CC0	MSIVPR6—Message shared interrupt vector/priority register 6	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1CD0	MSIDR6—Message shared interrupt destination register 6	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1CE0	MSIVPR7—Message shared interrupt vector/priority register 7	Mixed	0x1000_0000	10.3.6.4/10-38
0x5_1CF0	MSIDR7—Message shared interrupt destination register 7	Mixed	0x0000_0001	10.3.6.5/10-38
0x5_1D00– 0x5_FFF0	Reserved	—	—	—
<b>PIC Register Address Map—Per-CPU Registers</b>				
0x6_0000– 0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.8.1/10-45
0x6_0050	IPIDR1—Interprocessor interrupt 1 (IPI 1) dispatch register			
0x6_0060	IPIDR2—Interprocessor interrupt 2 (IPI 2) dispatch register			
0x6_0070	IPIDR3—Interprocessor interrupt 3 (IPI 3) dispatch register			
0x6_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.8.2/10-46
0x6_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.8.3/10-47
0x6_00A0	IACK—interrupt acknowledge register	R	0x0000_0000	10.3.8.4/10-47
0x6_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.8.5/10-48
<b>Global Utilities Registers</b>				
<b>Power-On Reset Configuration Values</b>				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00nn_n0nn	19.4.1.1/19-4
0xE_0004	PORBMSR—POR boot mode status register	R	0xn000_0000	19.4.1.2/19-6

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	Mixed	0x000n_007F	19.4.1.3/19-7
0xE_000C	PORDEVSR—POR I/O device status register	R	see ref.	19.4.1.4/19-8
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	19.4.1.5/19-10
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	19.4.1.7/19-11
<b>Signal Multiplexing and GPIO Controls</b>				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	19.4.1.8/19-11
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	19.4.1.9/19-12
0xE_0050	GPINDR—General-purpose input data register	R	0xnnnn_0000	19.4.1.10/19-13
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	19.4.1.11/19-13
<b>Device Disables</b>				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	19.4.1.12/19-14
<b>Power Management Registers</b>				
0xE_0080	POWMGTCSR—Power management status and control register	Mixed	0x0000_0000	19.4.1.13/19-16
<b>Interrupt and Reset Status and Control</b>				
0xE_0090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	19.4.1.14/19-17
0xE_0094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	19.4.1.15/19-18
<b>Version Registers</b>				
0xE_00A0	PVR—Processor version register	R	e500 processor version	19.4.1.16/19-18
0xE_00A4	SVR—System version register	R	system version	19.4.1.17/19-19
0xE_00B0	RSTCR—Reset control register	R/W	0x0000_0000	19.4.1.18/19-20
0xE_00C0	LBCVSELCR—LBC voltage select control register	R/W	0x0000_0000	19.4.1.19/19-20
<b>Status Registers</b>				
0xE_0B20	DDRCR—DDR calibration status register	R	0x0000_0000	19.4.1.20/19-21
0xE_0B24	DDRCR—DDR control driver register	R/W	0x0000_0000	19.4.1.21/19-21
0xE_0B28	DDRCLKDR—DDR clock disable register	R/W	0x0000_0000	19.4.1.22/19-22
<b>Debug Control</b>				
0xE_0E00	CLKOCR—Clock out control register	R/W	0x0000_0000	19.4.1.23/19-23
<b>Performance Monitor Control Registers</b>				
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	20.3.2.1/20-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	20.3.2.2/20-6
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	20.3.2.2/20-6
0xE_1018	PMC0 (lower)—Performance monitor counter 0 upper	R/W	0x0000_0000	20.3.3.1/20-10
0xE_101C	PMC0 (upper)—Performance monitor counter 0 lower	R/W	0x0000_0000	20.3.3.1/20-10

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
0xE_10A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_10A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	<a href="#">20.3.2.2/20-6</a>
0xE_10A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	<a href="#">20.3.3.1/20-10</a>
<b>Debug and Watchpoint Monitor Registers</b>				
<b>Watchpoint Monitor Registers</b>				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	<a href="#">21.1.3.1/21-4</a>
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	<a href="#">21.3.1.1/21-10</a>
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	<a href="#">21.3.1.2/21-12</a>
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	<a href="#">21.3.1.3/21-13</a>
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	<a href="#">21.3.1.4/21-13</a>
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	<a href="#">21.3.1.5/21-15</a>
<b>Trace Buffer Registers</b>				

Table B-1. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	<a href="#">21.3.2.1/21-15</a>
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	<a href="#">21.3.2.1/21-15</a>
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	<a href="#">21.3.2.2/21-18</a>
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	<a href="#">21.3.2.3/21-18</a>
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	<a href="#">21.3.2.4/21-19</a>
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	<a href="#">21.3.2.5/21-19</a>
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	<a href="#">21.3.2.6/21-20</a>
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	<a href="#">21.3.2.7/21-21</a>
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	<a href="#">21.3.2.8/21-21</a>
<b>Context ID Registers</b>				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	<a href="#">21.3.3.1/21-22</a>
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	<a href="#">21.3.3.2/21-23</a>

<sup>1</sup> Implementation-dependent reset values are listed in specified section/page.

<sup>2</sup> I<sup>2</sup>C2 has the same memory-mapped registers that are described for I<sup>2</sup>C1 from 0x000 to 0x014, except the offsets range from 0x100 to 0x114.

<sup>3</sup> Port size for BR0 is configured from external signals during reset, hence 'nn' is either 0x08, 0x10, or 0x18.

<sup>4</sup> Registers denoted \* are new to the enhanced TSEC and not supported by PowerQUICC III TSECs.

<sup>5</sup> Cleared on read.

<sup>6</sup> eTSEC3 has the same memory-mapped registers that are described for eTSEC1 from 0x\_2\_4000 to 0x2\_4FFF, except the offsets are from 0x\_2\_5000 to 0x2\_5FFF.

# Glossary

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this reference manual.

---

## A

**Architecture.** A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

**Atomic access.** A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The Power Architecture technology implements atomic accesses through the **lwarx/stwex** instruction pair.

**Autobaud.** The process of determining a serial data rate by timing the width of a single bit.

---

## B

**Beat.** A single state on the bus interface that may extend across multiple bus cycles. A transaction can be composed of multiple address or data *beats*.

**Big-endian.** A byte-ordering method in memory where the address *n* of a word corresponds to the *most-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the *most-significant byte*. See *Little endian*.

**Boundedly undefined.** A characteristic of certain operation results that are not rigidly prescribed by the Power Architecture technology. Boundedly-undefined results for a given operation may vary among implementations and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be boundedly undefined are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

**Breakpoint.** A programmable event that forces the core to take a breakpoint exception.

**Burst.** A multiple-beat data transfer whose total size is typically equal to a cache block.

**Bus clock.** Clock that causes the bus state transitions.



**Bus master.** The owner of the address or data bus; the device that initiates or requests the transaction.

---

**C**

**Cache.** High-speed memory containing recently accessed data or instructions (subset of main memory).

**Cache block.** A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache block may vary among processors; the maximum block size is one *page*. In Power Architecture processors, *cache coherency* is maintained on a cache-block basis. Note that the term ‘cache block’ is often used interchangeably with ‘cache line.’

**Cache coherency.** An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor’s cache.

**Cache flush.** An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

**Caching-inhibited.** A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

**Cast out.** A *cache block* that must be written to memory when a cache miss causes a cache block to be replaced.

**Changed bit.** One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. See also *Page access history bits* and *Referenced bit*.

**Clean.** An operation that causes a cache block to be written to memory, if modified, and then left in a valid, unmodified state in the cache.

**Clear.** To cause a bit or bit field to register a value of zero. See also *Set*.

**Context synchronization.** An operation that ensures that all instructions in execution complete past the point where they can produce an *exception*, that all instructions in execution complete in the context in which they began execution, and that all subsequent instructions are *fetched* and executed in the new context. Context synchronization may result from executing specific instructions (such as **isync** or **rfi**) or when certain events occur (such as an exception).

**Copy-back operation.** A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.



- 
- D**
- Direct-mapped cache.** A cache in which each main memory address can appear in only one location within the cache; operates more quickly when the memory request is a cache hit.
- Double data rate.** Memory that allows data transfers at the start and end of a clock cycle, thereby doubling the data rate.
- 
- E**
- Effective address (EA).** The 32-bit address specified for a load, store, or an instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* address or an I/O address.
- Exclusive state.** MEI state (E) in which only one caching device contains data that is also in system memory.
- 
- F**
- Fetch.** Retrieving instructions from either the cache or main memory and placing them into the instruction queue.
- Flush.** An operation that causes a cache block to be invalidated and the data, if modified, to be written to memory.
- Frame-check sequence (FCS).** Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD, and CRC.
- 
- G**
- General-purpose register (GPR).** Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.
- Guarded.** The guarded attribute pertains to out-of-order execution. When a page is designated as guarded, instructions and data cannot be accessed out-of-order.
- 
- H**
- Harvard architecture.** An architectural model featuring separate caches and other memory management resources for instructions and data.
- 
- I**
- Illegal instructions.** A class of instructions that are not implemented for a particular processor. These include instructions not defined by the architecture. In addition, for 32-bit implementations, instructions that are defined only for 64-bit implementations are considered to be illegal instructions. For 64-bit implementations instructions that are defined only for 32-bit implementations are considered to be illegal instructions.

**Implementation.** A particular processor that conforms to the architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features.

**Inbound ATMU windows.** Mappings that perform address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and map the transaction to its target interface.

**In-order.** An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model.

**Integer unit.** An execution unit in the core responsible for executing integer instructions.

**Inter-packet gap.** The gap between the end of one Ethernet packet and the beginning of the next transmitted packet.

**Instruction latency.** The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

---

**K** **Kill.** An operation that causes a *cache block* to be invalidated without writing any modified data to memory.

---

**L** **L2 cache.** Level-2 cache. See *Secondary cache*.

**Latency.** The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

**Least-significant bit (lsb).** The bit of least value in an address, register, field, data element, or instruction encoding.

**Least-significant byte (LSB).** The byte of least value in an address, register, data element, or instruction encoding.

**Little-endian.** A byte-ordering method in memory where the address *n* of a word corresponds to the *least-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *most-significant byte*. See *Big endian*.

**Local access window.** Mapping used to translate a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The local memory map is defined by a set of eight local access windows. The size of each window can be configured from 4 Kbytes to 2 Gbytes.

---

**M** **Media access control (MAC) sublayer.** Sublayer that provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.

- Media-independent interface (MII) sublayer.** Sublayer that provides a standard interface between the MAC layer and the physical layer for 10/100-Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.
- Medium-dependent interface (MDI) sublayer.** Sublayer that defines different connector types for different physical media and PMD devices.
- Memory access ordering.** The specific order in which the processor performs load and store memory accesses and the order in which those accesses complete.
- Memory-mapped accesses.** Accesses whose addresses use the page or block address translation mechanisms provided by the MMU and that occur externally with the bus protocol defined for memory.
- Memory coherency.** An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.
- Memory consistency.** Refers to agreement of levels of memory with respect to a single processor and system memory (for example, on-chip cache, secondary cache, and system memory).
- Memory management unit (MMU).** The functional unit that is capable of translating an *effective (logical) address* to a physical address, providing protection mechanisms, and defining caching methods.
- Modified/exclusive/invalid (MEI).** *Cache coherency* protocol used to manage caches on different devices that share a memory system. Note that neither the PowerPC ISA nor the Power ISA definitions specifies the implementation of an MEI protocol to ensure cache coherency.
- Modified state.** MEI state (M) in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.
- Most-significant bit (msb).** The highest-order bit in an address, registers, data element, or instruction encoding.
- Most-significant byte (MSB).** The highest-order byte in an address, registers, data element, or instruction encoding.

---

**N**

- NaN.** An abbreviation for not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs—signaling NaNs and quiet NaNs.
- No-op.** No-operation. A single-cycle operation that does not affect registers or generate bus activity.

- 
- O**
- OCeaN.** (On-chip network) Non-blocking crossbar switch fabric. Enables full duplex port connections at 128Gb/s concurrent throughput and independent per port transaction queuing and flow control. Permits high bandwidth, high performance, as well as the execution of multiple data transactions.
- Outbound ATMU windows.** Mappings that perform address translations from local 32-bit address space to the address spaces of, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.
- 
- P**
- Packet.** A unit of binary data that can be routed through a network. Sometimes packet is used to refer to the frame plus the preamble and start frame delimiter (SFD).
- Page.** A region in memory. The OEA defines a page as a 4-Kbyte area of memory aligned on a 4-Kbyte boundary.
- Page access history bits.** The *changed* and *referenced* bits in the PTE keep track of the access history within the page. The referenced bit is set by the MMU whenever the page is accessed for a read or write operation. The changed bit is set when the page is stored into. See [Changed bit](#) and [Referenced bit](#).
- Page fault.** A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory*. A page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.
- Page table.** A table in memory is comprised of *page table entries*, or PTEs. It is further organized into eight PTEs per PTEG (page table entry group). The number of PTEGs in the page table depends on the size of the page table (as specified in the SDR1 register).
- Page table entry (PTE).** Data structures containing information used to translate [effective address](#) to physical address on a 4-Kbyte page basis. A PTE consists of 8 bytes of information in a 32-bit processor and 16 bytes of information in a 64-bit processor.
- Physical coding sublayer (PCS).** Sublayer responsible for encoding and decoding data stream to and from the MAC sublayer.
- Physical medium attachment (PMA) sublayer.** Sublayer responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SERDES) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers.
- Physical medium dependent (PMD) sublayer.** Sublayer responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.

**Physical memory.** The actual memory that can be accessed through the system's memory bus.

**Pipelining.** A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

**Primary opcode.** The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction.

**Program order.** The order of instructions in an executing program. More specifically, this term is used to refer to the original order in which program instructions are fetched into the instruction queue from the cache.

**Protection boundary.** A boundary between *protection domains*.

**Protection domain.** A protection domain is a segment, a virtual page, a BAT area, or a range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

---

## Q

**Quad word.** A group of 16 contiguous locations starting at an address divisible by 16.

**Quiesce.** To come to rest. The processor is said to quiesce when an exception is taken or a **sync** instruction is executed. The instruction stream is stopped at the decode stage and executing instructions are allowed to complete to create a controlled context for instructions that may be affected by out-of-order, parallel execution. See [Context synchronization](#).

---

## R

**rA.** The rA instruction field is used to specify a GPR to be used as a source or destination.

**rB.** The rB instruction field is used to specify a GPR to be used as a source.

**rD.** The rD instruction field is used to specify a GPR to be used as a destination.

**rS.** The rS instruction field is used to specify a GPR to be used as a source.

**Record bit.** Bit 31 (or the Rc bit) in the instruction encoding. When it is set, updates the condition register (CR) to reflect the result of the operation.

**Reconciliation sublayer.** Sublayer that maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.

**Reduced instruction set computing (RISC).** An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

**Referenced bit.** One of two *page history bits* found in each *page table entry*. The processor sets the *referenced bit* whenever the page is accessed for a read or write. See also [Page access history bits](#).

**Reservation.** The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.

**Reservation station.** A buffer between the dispatch and execute stages that allows instructions to be dispatched even though the results of instructions on which the dispatched instruction may depend are not available.

---

## S

**Secondary cache.** A cache memory that is typically larger and has a longer access time than the primary cache. A secondary cache may be shared by multiple devices. Also referred to as L2, or level-2, cache.

**Set (v).** To write a nonzero value to a bit or bit field; the opposite of *clear*. The term ‘set’ may also be used to generally describe the updating of a bit or bit field.

**Set (n).** A subdivision of a *cache*. Cacheable data can be stored in a given location in one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was used least recently. See *Set associative*.

**Set-associative.** Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

**Slave.** The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

**Snooping.** Monitoring addresses driven by a bus master to detect the need for coherency actions.

**Snoop push.** Response to a snooped transaction that hits a modified cache block. The cache block is written to memory and made available to the snooping device.

**Stall.** An occurrence when an instruction cannot proceed to the next stage.

**Sticky bit.** A bit that when *set* must be cleared explicitly.

**Superscalar machine.** A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.

**Supervisor mode.** The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

**Synchronization.** A process to ensure that operations occur strictly *in order*. See *Context synchronization*.

**System memory.** The physical memory available to a processor.

- 
- T**
- Tenure.** The period of bus mastership. There can be separate address bus tenures and data bus tenures.
- Throughput.** The measure of the number of instructions that are processed per clock cycle.
- Time-division multiplex (TDM).** A single serial channel used by several channels taking turns.
- Transaction.** A complete exchange between two bus devices. A transaction is typically comprised of an address tenure and one or more data tenures, which may overlap or occur separately from the address tenure. A transaction may be minimally comprised of an address tenure only.
- Transfer termination.** Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure.
- Translation lookaside buffer (TLB).** A cache that holds recently-used *page table entries*.
- 
- U**
- User mode.** The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

- 
- V**
- Virtual address.** An intermediate address used in the translation of an *effective address* to a physical address.
- Virtual memory.** The address space created using the memory management facilities of the processor. Program access to *virtual memory* is possible only when it coincides with *physical memory*.
- 
- W**
- Way.** A location in the cache that holds a cache block, its tags, and status bits.
- Word.** A 32-bit data element.
- Write-back.** A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache block is *cast out* to make room for newer data.
- Write-through.** A cache memory update policy in which all processor write cycles are written to both the cache and memory.



# Index

## A

Accumulator (ACC), 6-47

Address maps

- addressing on PCI/PCI-X bus, 17-47

Address mask (LBC), 14-12

Address multiplexing (LBC SDRAM), 14-50

Address translation and mapping units (ATMUs)

- inbound windows, 2-9
  - illegal interactions between inbound ATMUs and local access windows, 2-10
  - PCI Express, 18-24
    - endpoint (EP) mode, 18-24
    - root complex (RC) mode, 18-25
  - PCI/PCI-X—4 windows, 2-10, 17-19
- local access windows, 2-3–2-10
  - see also* Local access windows
- outbound windows, 2-9
  - PCI Express, 18-19
  - PCI/PCI-X—4 windows, 17-16

Addressing

- PCI bus addressing, 17-48
  - configuration space, 17-48
  - I/O space, 17-48
  - memory space, 17-47

Alignment, byte (PCI/PCI-X), 17-49

Arbitration

- I<sup>2</sup>C interface
  - arbitration control, 11-15
  - loss of arbitration—forcing of slave mode, 11-24
  - procedure for arbitration, 11-15
- PCI/PCI-X interface, 17-5, 17-42

Architecture

- overview, 1-8

ASLEEP (global utilities asleep) signal, 19-2, 19-28

ATMUs, *see* Address translation and mapping units

## B

Battery backup, self-refresh mode, 9-72

BBEAR (branch buffer address register), *see* e500 core, registers

BBTAR (branch buffer target address register), *see* e500 core, registers

Block diagrams

- DDR controller, 9-1, 9-41
- debug modes, watchpoint monitor, and trace buffer, 21-1
- DMA controller, 16-1
- DUART, 13-2
- e500 coherency module (ECM), 8-1

- eTSEC, 15-2
- I<sup>2</sup>C interface, 11-1
- interrupt controller (PIC), 10-47
- L2 cache/SRAM, 7-1
- local bus controller (LBC), 14-1
- PCI Express, 18-2
- PCI/PCI-X controller, 17-2
- performance monitor, 20-2
- security engine (SEC), 12-3

Boot mode

- CPU holdoff (POR), 4-15, 8-4
- POR status register (PORBMSR), 19-6

Boot page translation, 4-7

Boot ROM location (POR), 4-13

Boot sequencer

- boot holdoff mode (POR), 4-16, 8-4
- boot page translation, 4-7
- I<sup>2</sup>C interface, 11-2, 11-17–11-20
- overview, 1-17, 4-8
- POR configuration, 4-16

BUCSR (branch unit control and status register), *see* e500 core, registers

Buffer descriptors, *see* eTSEC, buffer descriptors

Burst operations (PCI)

- see* PCI/PCI-X controller, bus protocol

Bus operations

- PCI/PCI-X, *see* PCI/PCI-X controller, bus protocol

Byte alignment (PCI/PCI-X), 17-49

## C

Chaining

- performance monitor events, 20-27

CKSTP\_IN (global utilities checkstop in) signal, 19-2

CKSTP\_OUT (global utilities checkstop out) signal, 19-3

CLK\_OUT (global utilities clock out) signal, 19-3, 19-23

Clocks

- DDR clock distribution, 9-57
- DDR controller clock disable, 19-22
- device clock signals summary, 4-3
  - see also* Signals, clock
- device clocking operation, 4-21–4-24
  - CCB (platform) clock, 4-21
    - minimum CCB frequency equations, 4-22
  - Ethernet clocks, 4-23
  - RapidIO clocks, 4-22
  - system clock/PCI clock, 4-21

eTSEC

- inputs and outputs, 15-9
- management clock out (EC\_MDC), 15-10, 15-73

I<sup>2</sup>C  
 clock stretching, 11-17  
 clock synchronization, 11-16  
 input synchronization and digital filter, 11-16  
 LBC bus clocks and clock ratios, 14-3  
 clock ratio register (LCRR), 14-30  
 overview, 1-20  
 PCI/PCI-X clocking, 17-45, 17-50  
 POR settings  
 e500 core PLL ratio, 4-12  
 system/CCB PLL ratio, 4-11  
 Coherency rules  
 L2 cache, 7-28  
 Commands  
 PCI, *see* PCI/PCI-X controller  
 Configuration  
 DDR, 9-11–9-32, 9-43  
 ECM  
 CCB address configuration register (EEBACR), 8-3  
 CCB port configuration register (EEBPCR), 8-4  
 eTSEC interfaces, 15-180–15-206  
 LBC  
 configuration register (LBCR), 14-29  
 SDRAM configurations supported, 14-47  
 PCI/PCI-X  
 configuration access registers, 17-60  
 configuration cycles, 17-58  
 configuration space header, 17-58  
 PIC  
 global configuration register, 10-19  
 POR, *see* Power-on-reset (POR)  
 Configuration space  
 PCI Express, 18-42  
 PCI/PCI-X addressing, 17-48  
 Configuration, control, and status  
 accessing CCSR memory from external masters, 2-10, 2-11  
 accessing CCSRs, 4-4  
 alternate configuration space (ALTBAR and ALTCAR),  
 4-6  
 boot page translation, 4-7  
 CCSR memory map, 2-10–2-14  
 CCSRBAR update guidelines, 4-4  
 memory map/register definition, 4-4  
 organization of CCSR memory, 2-11  
 Context ID registers, 21-23–21-24  
 CR (condition register), *see* e500 core, registers  
 Crypto-channels, *see* Security engine (SEC)  
 CSRR0–1 (critical save/restore registers 0–1), *see* e500 core,  
 registers  
 CTR (count register), *see* e500 core, registers  
 CTS, *see* DUART\_CTS[0:1]

## D

DACn (data address compare registers 1–2), *see* e500 core,  
 registers  
 Data cache  
*see* L2 cache/SRAM  
 DBCRn (debug control register 0–2), *see* e500 core, registers  
 DBSR (debug status register), *see* e500 core, registers  
 DDR controller  
 address signal mappings, 9-4  
 battery-backed RAM and forced self-refresh mode, 9-72  
 block diagram, 9-1, 9-41  
 clock distribution, 9-57  
 clocks  
 disabling, 19-22  
 configuration, example, 9-43  
 data beat ordering, 9-64  
 DDR2 calibration, 19-21  
 debug mode  
 signal selection (POR), 4-20  
 source and target ID, 21-4, 21-25  
 driver impedance calibration, 9-8  
 error checking and correcting (ECC), 7-39, 9-65  
 testing ECC with error injection, 9-33–9-34  
 error handling, 9-34, 9-67  
 features, 9-2  
 functional description, 9-41  
 I/O impedance control, 19-21  
 initialization/application information, 9-68  
 programming different memory types, 9-69  
 interrupts, 9-37  
 memory map/register definition, 9-9  
 modes of operation, 9-3  
 on-die termination for CSs, 9-8  
 overview, 1-14  
 page mode and logical bank retention, 9-64  
 performance monitor events, 20-17  
 register descriptions, 9-11  
 by acronym, *see* Register Index  
 configuration registers, 9-11–9-32  
 error handling registers, 9-34–9-41  
 error injection registers, 9-33–9-34  
 SDRAM operation, 9-45  
 address multiplexing, 9-47  
 initialization sequence, 9-72  
 JEDEC standard interface commands, 9-52  
 mode-set command timing, 9-58  
 organizations supported, 9-45  
 refresh operation, 9-60  
 power-saving modes, 9-61  
 timing, 9-61  
 registered DIMM mode, 9-59  
 timing, 9-54

- write timing adjustments, 9-59
- self-refresh
  - operation in sleep mode, 9-63
  - using forced mode for battery backup, 9-72
- signals summary, 9-3
  - see also* Signals, DDR
- DEAR (data exception address register), *see* e500 core, registers
- Debug modes
  - and watchpoint monitor signals summary, 21-5
    - see also* Signals, debug
  - and watchpoint monitor/trace buffer block diagram, 21-1
  - DDR signal selection (POR)
    - ECC pins used for debug, 4-20
  - DDR source ID debug modes, 21-4, 21-25
    - source ID on debug signals, 21-26
    - source ID on ECC pins, 21-26
  - DDR/LBC signal selection (POR), 4-20
  - e500 core registers, 6-39–6-45
  - features, 21-3
  - functional description, 21-25
  - LBC source ID debug mode, 14-4, 21-4, 21-25
  - memory map/register definition, 21-9
  - modes of operation (set at POR), 21-3
  - overview, 21-1
  - PCI/PCI-X
    - source ID debug mode, 21-25
  - performance monitor events, 20-27
  - POR status (global utilities), 19-10
  - READY negation, 4-2
  - software debug
    - context ID registers, 21-23
    - trace buffer, *see* Trace buffer
    - watchpoint, *see* Watchpoint monitor
- DEC (decrementer register), *see* e500 core, registers
- DECAR (decrementer auto-reload register), *see* e500 core, registers
- DMA channel 2 and 3 signal select, 19-13
- DMA controller
  - block diagram, 16-1
  - channel operation, 16-25
    - bandwidth control, 16-32
    - channel abort, 16-32
    - channel state, 16-32
    - stride size and distance, 16-33
  - descriptor formats, 16-34
  - error handling, 16-33
  - features, 16-2
  - functional description, 16-25
  - interrupts, 16-10–16-13, 16-15, 16-20, 16-24, 16-33
  - limitations and restrictions, 16-37
  - memory map/register definition, 16-6
  - modes of operation, 16-2
    - basic mode transfer, 16-26
      - basic chaining mode, 16-27
      - basic chaining single-write start mode, 16-28
      - basic direct mode, 16-26
        - basic direct single-write start mode, 16-26
    - channel continue mode for cascading transfer chains, 16-31
      - basic channel continue mode, 16-31
      - extended mode, 16-32
    - extended DMA mode transfer, 16-28
      - extended chaining mode, 16-29
      - extended chaining single-write start mode, 16-29
      - extended direct mode, 16-28
        - extended direct single-write start mode, 16-28
    - external control mode transfer, 16-30
  - overview, 16-2
  - performance monitor events, 20-18
  - register descriptions, 16-9–16-25
    - by acronym, *see* Register Index
  - signal select—channel 2 and 3, 19-13, 19-33
  - signals summary, 16-5
    - see also* Signals, DMA controller
  - system considerations, 16-38
    - unusual scenarios, 16-40
      - DMA to configuration and control registers, 16-40
      - DMA to DUART, 16-41
      - DMA to e500 core, 16-40
      - DMA to Ethernet, 16-40
      - DMA to I2C, 16-40
  - transfer interfaces, 16-33
- DMA\_DACK[0:3] (DMA acknowledge) signals, 16-6
- DMA\_DDONE[0:3] (DMA done) signals, 16-6
- DMA\_DREQ[0:3] (DMA request) signals, 16-6
- Doze mode, 1-20, 19-28
  - see also* Global utilities, power management
- DUART
  - asynchronous communication bits, 13-1
    - parity bit, 13-21
    - START bit, 13-20
    - STOP bit, 13-21
  - baud-rate generator logic, 13-21
  - block diagram, 13-2
  - divisor latch access bit (ULCRn[DLAB]), 13-4, 13-12
  - error handling, 13-22
    - framing error, 13-9, 13-15, 13-21, 13-22
    - overrun error, 13-22
    - parity error, 13-22
  - errors detected, 13-2
  - features, 13-1
  - functional description, 13-19
  - initialization/application information, 13-24

- interrupts
    - interrupt control logic, 13-23
    - interrupt enable and control registers, 13-9–13-11
  - memory map/register definition, 13-4
  - modes of operation, 13-2
    - DMA mode selection, 13-23
    - FIFO mode, 13-22
      - interrupts, 13-23
    - local loopback mode, 13-22
  - overview, 1-17, 13-1
  - PC16450 UART compatibility, 13-1
  - performance monitor events, 20-27
  - register descriptions, 13-4, 13-6–13-19
    - by acronym, see Register Index
    - UART0 register offsets, 13-4
    - UART1 register offsets, 13-4
  - serial interface data format, 13-2
  - serial interface operation, 13-20–13-21
    - data transfer, 13-21
    - START bit, 13-20
    - STOP bit, 13-21
    - transaction protocol example, 13-20
  - signals summary, 13-3
    - see also Signals, DUART
- E**
- e500 coherency module (ECM)
    - block diagram, 8-1
    - CCB arbiter, 8-9
    - CCB interface, 8-10
    - configuration
      - CCB address configuration register (EEBACR), 8-3
      - CCB port configuration register (EEBPCR), 8-4
    - error handling
      - error handling registers, 8-6–8-9
    - features, 8-2
    - functional description, 8-9
    - global data multiplexor, 8-10
    - I/O arbiter, 8-9
    - initialization/application information, 8-10–8-11
    - interrupts
      - ECM error enable register (EEER), 8-7
    - memory map/register definition, 8-3
    - overview, 1-14, 8-2
    - performance monitor events, 20-19
    - register descriptions, 8-3
      - by acronym, see Register Index
    - transaction queue, 8-10
  - e500 core
    - boot mode (POR), 4-15
    - branch operations
      - registers, 6-9–6-11
    - branch target buffer (BTB)
      - registers, 6-23–6-25
    - computational operations
      - registers, 6-8–6-9
    - debug registers, 6-39–6-45
    - hardware implementation-dependent registers (HID0–1), 6-25–6-28
    - interrupts
      - registers, 6-17–6-22
      - sources, 10-3
    - L1 caches
      - registers, 6-28–6-32
    - memory management unit (MMU)
      - registers, 6-32–6-39
        - MMU assist registers (MAS0–MAS4, MAS6), 6-35–6-38
        - MMU assist registers (MAS0–MAS4, MAS6–MAS7), 6-35–6-39
    - overview, 1-8
    - performance monitor
      - registers, 6-48–6-52
    - processor control registers, 6-11–6-14
    - registers
      - BBEAR (branch buffer address register), 6-23
      - BBTAR (branch buffer target address register), 6-23
      - BUCSR (branch unit control and status register), 6-24
      - CR (condition register), 6-9
      - CSRR0–1 (critical save/restore registers 0–1), 6-17
      - CTR (count register), 6-11
      - DACn (data address compare registers 1–2), 6-45
      - DBCRn (debug control register 0–2), 6-39–6-43
      - DBSR (debug status register), 6-43
      - DEAR (data exception address register 0), 6-18
      - DEC (decrementer register), 6-16
      - DECAR (decrementer auto-reload register), 6-17
      - ESR (exception syndrome register), 6-19
      - GPRs (general-purpose registers), 6-8
      - HID0–1 (hardware implementation-dependent registers 0–1), 6-25
      - IACn (instruction address compare registers 1–2), 6-45
      - IVORn (interrupt vector offset registers), 6-18
      - IVPR (interrupt vector prefix register), 6-18
      - L1CFG0–1 (L1 cache configuration registers 0–1), 6-30
      - L1CSR0–1 (L1 cache status and control registers 0–1), 6-28
      - LR (link register), 6-11
      - MAS0–MAS6 (MMU assist registers 0–6), 6-35–6-39
      - MCAR (machine check address register), 6-21
      - MCSR (machine check syndrome register), 6-21
      - MCSRR0–1 (machine check save/restore registers 0–1), 6-20
      - MMUCFG (MMU configuration register), 6-32

- MMUCSR0 (MMU control and status register 0), 6-32
- MSR (machine state register), 6-11
- PIDn (process ID registers 0–2), 6-32
- PIR (processor ID register), 6-13
- PMcN (performance monitor counter registers 0–3), 6-52
- PMGC0 (performance monitor global control register 0), 6-49
- PMLCan (performance monitor local control registers a0–a3), 6-50
- PMLCbn (performance monitor local control registers b0–b3), 6-51
- PVR (processor version register), 6-13
- SPEFSCR (signal processing and embedded floating-point status and control register), 6-45
- SPRGn (software-use registers 0–7), 6-22
- SRR0–1 (save/restore registers 0–1), 6-17
- SVR (system version register), 6-14, 19-19
- TBL (time base lower register), 6-16
- TBU (time base upper register), 6-16
- TCR (timer control register), 6-14
- TLB0CFG (TLB0 configuration register), 6-33
- TLB1CFG (TLB1 configuration register), 6-34
- TSR (timer status register), 6-15
- UPMcN (user performance monitor counter registers 0–3), 6-52
- UPMGC0 (user performance monitor global control register 0), 6-49
- UPMLCan (user performance monitor local control registers a0–a3), 6-50
- UPMLCbn (user performance monitor local control registers b0–b3), 6-51
- USPRG0 (user software-use register 0), 6-22
- XER (integer exception register), 6-8
- signal processing engine (SPE)
  - registers, 6-45
- software-use SPRs, 6-22
- time base
  - RTC (real time clock) signal options, 4-3, 4-23
  - timer registers, 6-14–6-17
- EC\_GTX\_CLK125 (eTSEC gigabit transmit 125 MHz source) signal, 15-10
- EC\_MDC (eTSEC management data clock) signal, 15-10
- EC\_MDIO (eTSEC management data input/output, BIDI) signal, 15-10
- Encryption algorithms, *see* Security engine (SEC), execution units (EUs)
- Error handling
  - DDR, 9-34–9-41, 9-67
  - DMA, 16-33
  - DUART, 13-2, 13-22
  - framing error, 13-9, 13-15, 13-21, 13-22
  - overrun error, 13-22
  - parity error, 13-22
- ECM
  - error handling registers, 8-6–8-9
- eTSEC, 15-155–15-157
- I<sup>2</sup>C interface
  - boot sequencer mode, 11-19
- L2 cache/SRAM
  - error handling registers, 7-17
  - error injection, 7-18
- LBC
  - transfer error registers, 14-24–14-29
- PCI Express registers, 18-29–18-42
- PCI/PCI-X
  - address/data parity, 17-54, 17-65, 17-66
  - reporting, 17-65–17-66
    - PERR and SERR signals, 17-66
    - target-initiated termination, 17-53
  - retry transactions, 17-53
  - target-abort, 17-53
  - target-disconnect, 17-53
  - security engine (SEC), 12-98, 12-103
- ESR (exception syndrome register), *see* e500 core, registers
- eTSEC
  - block diagram, 15-2
  - buffer descriptors, 15-173–15-180
    - receive buffer descriptors (RxBD), 15-178
    - transmit buffer descriptors (TxBD), 15-174
  - clocks
    - inputs and outputs, 15-9
    - management clock out (EC\_MDC), 15-10, 15-73
    - operation, 4-23
  - configuration of interfaces, 15-180–15-206
  - 8-bit FIFO mode, 15-205
  - GMII interface mode, 15-185
  - MAC configuration, 15-64
  - MII interface mode, 15-181
  - RGMIII interface mode, 15-193
  - RMII interface mode, 15-197
  - RTBI interface mode, 15-201
  - TBI interface mode, 15-189
  - data width (POR), 4-17
  - error-handling, 15-155–15-157
  - eTSEC1 protocol (POR), 4-18
  - eTSEC3 protocol (POR), 4-18
  - features, 15-2
  - FIFO interface connections, 15-137
    - 8-bit encoded packet FIFO mode, 15-140
    - 8-bit GMII-style packet FIFO mode, 15-139
    - CRC appending and checking, 15-138
    - flow control, 15-138
    - signal summary, 15-140

- functional description, 15-126
  - gigabit Ethernet channel operation, 15-141
    - flow control, 15-151
    - frame reception, 15-144
    - frame recognition, 15-147
    - frame transmission, 15-143
    - initialization sequence, 15-141
      - soft reset and reconfiguring procedure, 15-142
    - internal and external loop back, 15-155
    - inter-packet gap time, 15-155
    - Magic Packet mode, 15-151
    - preamble customization, 15-145
    - RMON support, 15-147
  - hash function
    - algorithm, 15-149
    - registers, 15-107–15-108
  - initialization/application information, 15-180–15-206
    - gigabit Ethernet channel, 15-141
      - soft reset and reconfiguring procedure, 15-142
    - see also* eTSEC, configuration
  - interrupts, 15-152–15-155
    - interrupt coalescing, 15-153
      - by frame count threshold, 15-153
      - by timer threshold, 15-154
    - interrupt registers, 15-24–15-29
  - lossless flow control, 15-170
    - back pressure determination and free buffers, 15-170
    - software use of hardware-initiated back pressure, 15-172
  - MAC functionality, 15-64–15-79
    - configuration, 15-64
      - CSMA/CD control, 15-64
      - handling packet collisions, 15-64
      - packet flow control, 15-65
      - PHY links control, 15-66
      - registers, 15-66–15-79
  - memory map/register definition, 15-12
    - detailed memory map, 15-13–15-22
    - eTSEC2–4 controller offsets, 15-22, B-40
    - top-level module map, 15-13
  - modes of operation, 15-4
    - RMON support, 15-79
  - overview, 1-18, 15-1
  - physical interface connections, 15-126
    - gigabit media-independent interface (GMII), 15-128
    - media-independent interface (MII), 15-127
    - reduced gigabit media-independent interface (RGMII), 15-129
    - reduced media-independent interface (RMII), 15-127
    - reduced ten-bit interface (RTBI), 15-132
    - ten-bit interface (TBI), 15-131
  - quality of service (QoS) support, 15-162–15-170
    - receive queue filer, 15-162
    - transmission scheduling, 15-168
  - register descriptions, 15-22–15-126
    - by acronym, *see* Register Index
    - DMA attribute registers, 15-110–15-112
    - FIFO registers, 15-109–15-110
    - general control and status registers, 15-22–15-36
    - hash function registers, 15-107–15-108
    - lossless flow control registers, 15-112–15-114
    - MAC registers, 15-66–15-79
    - MIB registers, 15-79–15-107
    - receive control and status registers, 15-48–15-63
    - ten-bit interface registers, 15-115–15-126
    - transmit control and status registers, 15-36–15-48
  - signals, 15-6–15-12
    - FIFO interface signal summary, 15-140
    - see also* Signals, eTSEC
    - summary, 15-6
    - TCP/IP off-load, 15-157–15-162
      - frame control blocks, 15-158
      - receive path off-load, 15-160
      - transmit path off-load, 15-158
  - eTSEC2 signals as GP I/O, *see* Global utilities, general-purpose I/O signals
  - External system configuration
    - POR (LAD[0:31]) status, 4-21, 19-11
  - External writes, *see* L2 cache/SRAM, stashing
- ## F
- Full-on mode (power), 1-20
- ## G
- General-purpose I/O (PCI and eTSEC2)
    - see* Global utilities
  - Global utilities
    - clock out
      - CLK\_OUT signal, 19-3, 19-23
      - clock out control register (CLKOCR), 19-23
      - overview, 19-1
    - DDR calibration status, 19-21
    - DDR controller
      - clock disable, 19-22
    - DMA signal multiplex control register (PMUXCR), 19-13, 19-33
    - features, 19-1
    - functional description, 19-25
    - general-purpose I/O signals (PCI and eTSEC2)
      - control register (GPIOCR), 19-11
      - GPOUT[24:31] signals, 19-3
      - input data register (GPINDR), 19-13
      - operation of, 19-33
      - output data register (GPOUTDR), 19-12

- I/O impedance
    - DDR controller, software select, 19-21
  - interrupt and local bus signal multiplexing, 19-1, 19-13
    - operation, 19-33
  - interrupts and power management, 19-31
  - LBC voltage select, 19-20
  - machine check summary
    - sources of *mcp* (MCPSUMR), 19-17
  - memory map/register definition, 19-3
  - overview, 19-1
  - POR configuration
    - boot mode status register (PORBMSR), 19-6
    - debug mode status register (PORDBGMSR), 19-10
    - device status register (PORDEVSR), 19-8
    - I/O impedance status register (PORIMPSCR), 19-7
    - LAD[0:31] external system configuration (GPPORCR), 4-21, 19-11
    - PLL status register (PORPLLSR), 19-4
      - see also* Power-on reset (POR)
  - power management
    - and interrupts, 19-31
    - and snooping, 19-32
    - block disable (DEVDISR), 19-14, 19-27
    - CKSTP\_IN and core\_stopped mode, 19-27
    - core and device control bits, 19-28
    - core and device modes, 19-26
    - device mode control and status register (POWMGTCSR), 19-16
    - doze mode, 19-28
    - dynamic power management, 19-27
    - features, 19-1
    - functional description, 19-25
    - nap mode, 19-28
    - power-down sequence, 19-29
    - sleep mode, 19-28, 19-32
    - software considerations, 19-32
  - processor version register (PVR), 19-18
  - register descriptions, 19-4
    - by acronym, *see* Register Index
  - reset
    - HRESET\_REQ control, 19-20
    - RapidIO and PCI Express reset requests (RSTRSCR), 19-18
  - signals summary, 19-2
    - see also* Signals, global utilities
  - snooping and power management, 19-32
  - system version register (SVR), 19-19
  - GPCM (LBC general-purpose chip-select machine), 14-36
    - see also* Local bus controller (LBC)
  - GPOUT[24:31] (general-purpose outputs) signal, 19-3
  - GPRs (general-purpose registers), *see* e500 core, registers
- ## H
- Hash function, *see* eTSEC, hash function
  - HID0–1 (hardware implementation-dependent registers 0–1), *see* e500 core, registers
  - HRESET (hard reset) signal, 4-2, 4-8
  - HRESET\_REQ (hard reset request) signal, 4-2, 11-18, 11-19
- ## I
- I/O impedance
    - DDR controller driver select, 19-21
    - LBC and PCI/PCI-X signals
      - control and status register (global utilities), 19-7
      - PCI/PCI-X interface (POR), 4-20
  - I/O requirements, 19-32
  - I/O space
    - PCI/PCI-X addressing, 17-48
  - I<sup>2</sup>C interface
    - arbitration
      - arbitration control, 11-15
      - loss of arbitration—forcing of slave mode, 11-24
      - procedure for arbitration, 11-15
    - block diagram, 11-1
    - boot sequencer
      - POR configuration, 4-16
    - boot sequencer mode, 11-2, 11-17–11-20
      - error condition behavior, 11-19
    - calling address match condition, 11-6
    - clock control, 11-16
      - clock stretching, 11-17
      - clock synchronization, 11-16
      - input synchronization and digital filter, 11-16
      - master mode, 11-16
      - slave mode, 11-16
    - data transfer, 11-13
    - error handling
      - boot sequencer mode, 11-19
    - features, 11-2
    - frequency divider
      - frequency divider register (I2CFDR), 11-6
    - functional description, 11-11
    - handshaking, 11-16
    - implementation details, 11-14
      - address compare, 11-15
      - control transfer, 11-14
      - transaction monitoring, 11-14
    - initialization/application information, 11-21–11-25
      - boot sequencer mode, *see* I<sup>2</sup>C interface, boot sequencer mode
      - generation of SCL when SDA low, 11-23
      - initialization sequence, 11-21
      - post-transfer software response, 11-22

- repeated START generation, 11-23
- START generation, 11-12, 11-22
- STOP generation, 11-13, 11-23
- interrupts
  - calling address match condition, 11-6
  - flowchart for interrupt service routine, 11-24
  - interrupt after transfer, 11-22
  - interrupt enable bit (I2CCR[MIEN]), 11-8
  - interrupt on START, 11-22
  - interrupt pending status bit (I2CSR[MIF]), 11-10
  - interrupt-driven byte-to-byte transfers, 11-2
  - read of last byte, 11-23
  - slave mode interrupt service routine guidelines, 11-23
    - for slave transmitter routine, 11-24
    - loss of arbitration, 11-24
- memory map/register definition, 11-4
- modes of operation, 11-2
  - boot sequencer mode, 11-2, 11-17–11-20
  - interrupt-driven byte-to-byte data transfer, 11-2
  - master mode, 11-2
  - slave mode, 11-2
- overview, 11-2
- register descriptions, 11-5
  - by acronym, see Register Index
- signals summary, 11-3
  - see also Signals, I<sup>2</sup>C
- transaction protocol, 11-11
  - handshaking, 11-16
  - repeated START condition, 11-3, 11-13
  - slave address transmission, 11-12
  - START condition, 11-3, 11-12, 11-22
  - STOP condition, 11-3, 11-13, 11-23
- IACn (instruction address compare registers 1–2), see e500 core, registers
- Initialization
  - DDR (initialization and application information), 9-68
    - programming different memory types, 9-69
  - ECM (initialization and application information), 8-10–8-11
  - eTSEC (initialization and application information), 15-141, 15-180–15-206
    - see also eTSEC, configuration
  - I<sup>2</sup>C interface (initialization and application information), 11-21–11-25
    - boot sequencer mode, see I<sup>2</sup>C interface, boot sequencer mode
    - generation of SCL when SDA low, 11-23
    - initialization sequence, 11-21
    - post-transfer software response, 11-22
    - repeated START generation, 11-23
    - START generation, 11-12, 11-22
    - STOP generation, 11-13, 11-23
  - LBC (initialization and application information), 14-79–14-112
  - LBC SDRAM power-on initialization, 14-48
  - PCI/PCI-X (initialization and application information), 17-67–17-70
  - PIC (initialization and application information), 10-52
  - watchpoint monitor and trace buffer, 21-31
- Intel PC133 SDRAM commands (LBC), 14-49
- Interrupt controller (PIC)
  - block diagram, 10-47
  - configuration (global), 10-19
  - critical interrupts, 10-6, 10-27, 10-30, 10-39
  - destination (interrupt routing), 10-39
    - IRQ\_OUT, 10-6
      - see also e500 core, critical interrupts
  - end of interrupt (EOI), 10-47, 10-49
  - external interrupts
    - routed to critical interrupt (cint), 10-30
    - routed to IRQ\_OUT, 10-28
  - features, 10-3
  - flow (interrupt processing), 10-47
  - functional description, 10-47
  - global timers, 10-22, 10-51
    - cascading of timers, 10-25, 10-27
    - clocking of timers, 10-23, 10-27
    - RTC (real time clock) signal options, 4-3, 4-23, 10-25, 10-26
  - initialization/application information, 10-52
  - interrupt acknowledge (IACK) signaling, 10-46, 10-49
  - interrupt routing (mixed mode), 10-6, 10-47
  - interrupt source priorities, 10-49
  - memory map/register definition, 10-9
  - messaging interrupts, 10-50
  - modes of operation, 10-4, 10-19
    - mixed mode, 10-4
    - pass-through mode (to support external interrupt controllers), 10-5
  - nesting of interrupts, 10-50
  - overview, 1-15, 10-1
  - performance monitor events, 20-20
  - power management (wake up conditions), 10-3
  - processor core interrupt sources, 10-3
    - critical interrupt (cint) sources, 10-27
  - processor current task priority, 10-49
  - programming guidelines, 10-52
    - changing interrupt source configuration, 10-54
  - register descriptions, 10-17
    - by acronym, see Register Index, 10-17
    - global registers, 10-17–10-22
    - global timer registers, 10-22–10-27
    - interrupt source configuration registers, 10-22–10-25, 10-38–10-43



- message registers, 10-33–10-35
- non-accessible registers
  - in-service register (ISR), 10-49
  - interrupt pending register (IPR), 10-47
  - interrupt request register (IRR), 10-47
- per-CPU registers, 10-43–10-47
- performance monitor mask registers, 10-31–10-33
- summary registers, 10-27–10-31
- reset of PIC, 10-19, 10-52
- reset processor from software, 10-20
- signals summary, 10-7
  - see also Signals, PIC
- simultaneous interrupts, priorities, 10-49
- sources of interrupts, 10-5
  - internal (to PIC) interrupt destinations, 10-28, 10-29, 10-30, 10-31
  - internal (to PIC) interrupt sources, 10-6
- spurious vector generation, 10-22, 10-50
- vendor identification, 10-20
- Interrupts
  - DDR, 9-37
  - DMA, 16-10–16-13, 16-15, 16-20, 16-24, 16-33
  - DUART
    - interrupt control logic, 13-23
    - interrupt enable and control registers, 13-9–13-11
  - e500 core
    - registers, 6-17–6-22
  - ECM interrupt register (ECM error enable register—EEER), 8-7
  - eTSEC, 15-152–15-155
    - interrupt registers, 15-24–15-29
  - I<sup>2</sup>C interface
    - calling address match condition, 11-6
    - flowchart for interrupt service routine, 11-24
    - interrupt after transfer, 11-22
    - interrupt enable bit (I2CCR[MIEN]), 11-8
    - interrupt on START, 11-22
    - interrupt pending status bit (I2CSR[MIF]), 11-10
    - interrupt-driven byte-to-byte transfers, 11-2
    - read of last byte, 11-23
    - slave mode interrupt service routine guidelines, 11-23
      - for slave transmitter routine, 11-24
      - loss of arbitration, 11-24
  - IRQ[9:11] signal select, 19-13, 19-33
  - LBC interrupt register, 14-26
  - PCI/PCI-X error enable register, 17-26
  - performance monitor (PIC), 20-20
  - power management and interrupts (global utilities), 19-31
  - security engine (SEC), 12-103–12-104, 12-108
    - registers, 12-110–12-112
  - see also Interrupt controller (PIC)
  - IRQ[0:11] (interrupt request 0–11) signals, 10-8
  - IRQ[9:11] signal select
    - global utilities, 19-13
  - IRQ\_OUT (interrupt request out) signal, 10-8, 10-27
  - IVORn (interrupt vector offset registers), see e500 core, registers
  - IVPR (interrupt vector prefix register), see e500 core, registers
- J**
  - JEDEC SDRAM commands (LBC), 14-49
  - JTAG test access port
    - signals summary, 21-5
    - see also Signals, JTAG, 21-5
- L**
  - L1CFG0–1 (L1 cache configuration registers 0–1), see e500 core, registers
  - L1CSR0–1 (L1 cache status and control registers 0–1), see e500 core, registers
  - L2 cache/SRAM
    - allocation of lines, 7-32
    - block diagram, 7-1
    - coherency rules, 7-28
    - error handling registers, 7-17
    - error injection, 7-18
    - external writes, *see* stashing
    - flash clearing, instruction and data locks, 7-31
    - locking
      - clearing locks on selected lines, 7-30
      - entire, 7-29
      - programmed memory ranges, 7-30
      - selected lines, 7-30
      - with stale data, 7-31
    - memory map/register definition, 7-8
    - memory-mapped SRAM
      - coherency rules, 7-29
    - memory-mapped windows, 2-4
    - operation, 7-33
    - overview, 7-1
    - performance monitor events, 20-26
    - PLRU bit update considerations, 7-32
    - register descriptions, 7-10–7-25
    - replacement policy, 7-31
    - SRAM features, 7-2
    - stashing, 7-25
    - state transitions, 7-35
      - due to core-initiated transactions, 7-35
      - due to system-initiated transactions, 7-38
    - timing, 7-27
  - LA[27:31] (LBC non-multiplexed address) signals, 14-7
  - LAD[0:31] (LBC multiplexed address/data) signals, 14-7

- LALE (LBC external address latch enable) signal, 14-5, 14-33
- LBCTL (LBC data buffer control) signal, 14-7, 14-35
- LBS[0:3] (LBC UPM byte select) signals, 14-6
- LCK[0:2] (LBC clock) signals, 14-8
- LCKE (LBC clock enable) signal, 14-7
- LCPL[0:7] (LBC chip select) signals, 14-5
- $\overline{\text{LCS}}[5:7]$  signal select
  - global utilities, 19-13
- LCS0 (LBC chip select 0) signal, 14-46
- LDP[0:3] (LBC data parity) signals, 14-7, 14-36
- LGPL0 (LBC GP line 0) signal, 14-6
- LGPL1 (LBC GP line 1) signal, 14-6
- LGPL2 (LBC GP line 2) signal, 14-6
- LGPL3 (LBC GP line 3) signal, 14-6
- LGPL4 (LBC GP line 4) signal, 14-6
- LGPL5 (LBC GP line 5) signal, 14-7
- LGTA (LBC GPCM transfer acknowledge) signal, 14-6, 14-46
- Local access windows, 2-3–2-10
  - ATMUs, *see* Address translation and mapping units (ATMUs)
  - configuring local access windows, 2-8
  - distinguishing local access windows from other mapping functions, 2-9
  - illegal interactions
    - between inbound ATMUs and local access windows, 2-10
    - between local access windows and DDR SDRAM chip selects, 2-9
  - L2 cache/SRAM window interactions, 2-4
  - precedence if overlapping among themselves, 2-8
  - precedence if overlapping with L2 cache/SRAM windows, 2-4
  - registers, 2-7–2-8
    - by acronym, *see* Register Index
- Local address map, 1-20
  - see also* Local access windows
- Local bus controller (LBC)
  - address and address space checking, 14-33
  - address mask field—option registers, 14-12
  - atomic bus operations, 14-35
  - block diagram, 14-1
  - boot chip-select operation, 14-46
  - bus monitor, 14-36
  - bus turnaround, 14-82
    - additional address phases (UPM cycles), 14-83
    - address following read, 14-82
    - read data following address, 14-82
    - read-modify-write cycle (parity), 14-83
  - clocks and clock ratios, 14-3
    - clock ratio register (LCRR), 14-30
  - configuration
    - LBC configuration register (LBCR), 14-29
  - debug mode
    - signal selection (POR), 4-20
    - source and target ID, 21-4, 21-25
  - DSP hosts (interface to), 14-98
    - MSC8101 HDI16 interface, 14-98
    - MSC8102 DSI interface, 14-102
  - error handling
    - transfer error registers, 14-24–14-29
  - external access termination (LGTA), 14-46
  - features, 14-2
  - functional description, 14-32
  - general-purpose chip-select machine (GPCM), 14-36
    - chip-select and write enable negation timing, 14-40
    - chip-select assertion timing, 14-39
    - extended hold time on read accesses, 14-43
  - GPCM mode
    - registers, 14-13
  - output enable timing, 14-43
  - programmable wait state configuration, 14-40
  - relaxed timing, 14-41
  - timing configuration, 14-37
- initialization/application information, 14-79–14-112
- interrupts
  - transfer error interrupt enable register (LTEIR), 14-26
- $\overline{\text{LCS}}[5:7]$  signal select, 19-13
- LCS[5:7] signal select, 19-33
- memory map/register definition, 14-8
- memory refresh timer prescaler, 14-20
- modes of operation, 14-3
  - bus clock and clock ratios, 14-3
  - GPCM mode, registers, 14-13
  - power-down mode, 14-4
  - SDRAM mode, registers, 14-16
  - source ID debug mode, 14-4
  - UPM mode, registers, 14-15
- overview, 14-2
- parity generation and checking, 14-36, 14-95
- performance monitor events, 20-26
- peripherals, 14-79
  - GPCM timing, 14-81
  - hierarchy for very high speeds, 14-80
  - hierarchy on the local bus, 14-80
  - multiplexed address/data, 14-79
- port sizes, 14-83
- register descriptions, 14-10
  - by acronym, *see* Register Index
- SDRAM interface, 14-47–14-58, 14-85
  - address multiplexing, 14-50
  - basic capabilities, 14-85
  - commands (Intel PC133 and JEDEC), 14-49

- configurations supported, 14-47
  - device-specific parameters, 14-51
  - limitations, 14-87–14-95
  - maximum SDRAM supported, 14-86
  - page hit checking, 14-50
  - page management, 14-50
  - parity support, 14-95
  - power-on initialization, 14-48
  - refresh, 14-57
  - SDRAM mode
    - registers, 14-16, 14-21
  - timing, 14-55
    - activate-to-read/write interval, 14-52
    - CAS latency, 14-53
    - external buffers, 14-54
    - MODE-SET commands, 14-57
    - precharge-to-activate interval, 14-52
    - refresh recovery, 14-54
    - refresh timing, 14-58
    - write recovery, 14-53
  - transactions, 14-57
  - signals summary, 14-4
    - see also Signals, LBC
  - UPM interfaces, 14-58–14-78
    - block diagram, 14-59
    - example interface, 14-73
    - extended hold time (reads), 14-73
    - programming the UPMs, 14-62
  - RAM array, 14-64
    - address multiplexing, 14-70
    - byte select signal timing, 14-68
    - chip select signal timing, 14-67
    - data timing, 14-71
    - general purpose signal timing, 14-69
    - LGPL[0:5] timing (LAST), 14-71
    - loop control, 14-69
    - RAM word definition, 14-65
    - REDO, 14-69
    - wait mechanism (WAEN), 14-71
  - signal timing, 14-64
  - synchronous UPWAIT (early transfer acknowledge), 14-72
  - UPM mode
    - registers, 14-15, 14-17
  - UPM requests, 14-59
    - exception requests, 14-62
    - memory access requests, 14-60
    - refresh timer requests, 14-61
    - software requests, 14-61
  - voltage selection, 19-20
  - ZBT SRAM interface, 14-96
  - LOE (LBC GPCM output enable) signal, 14-6
  - LPBSE (LBC parity byte select) signal, 14-6
  - LR (link register), see e500 core, registers
  - LSDA10 (LBC SDRAM A10) signal, 14-6
  - LSDCAS (LBC SDRAM CAS) signal, 14-6
  - LSDDQM[0:3] (LBC SDRAM data mask) signal, 14-6
  - LSDRAS (LBC SDRAM RAS) signal, 14-6
  - LSDWE (LBC SDRAM write enable) signal, 14-6
  - LSYNC\_IN (LBC PLL synchronization in) signal, 14-8
  - LSYNC\_OUT (LBC PLL synchronization out) signal, 14-8
  - LWE[0:3] (LBC GPCM write enable) signals, 14-6
- ## M
- MA[0:14] (DDR address bus) signals, 9-7
  - MAC functionality, *see* eTSEC, MAC functionality
  - Machine check
    - MCP (processor machine check) signal, 10-8
    - mcp* summary register (MCPSUMR), 19-17
    - SRESET (soft reset) signal, 4-8
  - MAS0–MAS6 (MMU assist registers 0–6), *see* e500 core, registers
  - MBA[0:1] (DDR logical bank address) signals, 9-7
  - MCAR (machine check address register), *see* e500 core, registers
  - MCAS (DDR column address strobe) signal, 9-7
  - MCK[0:5] (DDR clock output complement) signals, 9-9
  - MCK[0:5] (DDR clock output) signals, 9-9
  - MCKE[0:3] (DDR clock enable) signals, 9-9
  - MCP (processor machine check) signal, 10-8
  - MCS[0:3] (DDR chip select) signals, 9-8
  - MCSR (machine check syndrome register), *see* e500 core, registers
  - MCSRR0–1 (machine check save/restore registers 0–1), *see* e500 core, registers
  - MDIC[0:1] (DDR driver impedance calibration) signals, 9-8
  - MDM[0:8] (DDR SDRAM data output mask) signals, 9-8
  - MDQ[0:8] (DDR data bus strobe) signals, 9-6, 9-42
  - MDVAL (DDR/LBC debug mode data valid) signal, 4-20, 14-8, 21-3, 21-6
  - MECC[0:5] (DDR error correcting code) signals as debug, 21-3, 21-7
  - MECC[0:7] (DDR error correcting code) signals, 4-20, 9-6
  - Memory maps
    - CCSR memory, 2-4
      - accessing CCSR memory from external masters, 2-10, 2-11
    - CCSR map, complete list of memory-mapped registers (by offset), 2-14
    - CCSR organization, 2-11
    - CCSR registers, 2-10–2-14
    - device-specific utilities, 2-13
    - general utilities registers, 2-12
    - programmable interrupt controller (PIC) space, 2-13

configuration, control, and status registers, 4-4  
 DDR controller, 9-9  
   illegal interaction between local access windows and  
   DDR SDRAM chip selects, 2-9  
 debug, watchpoint, and trace buffer registers, 21-9  
 device memory map  
   address translation and mapping, 2-3  
   overview and example, 2-1  
 DMA, 16-6  
 DUART, 13-4  
 ECM, 8-3  
 eTSEC, 15-12  
 global utilities, 19-3  
 I<sup>2</sup>C, 11-4  
 interrupt controller (PIC), 10-9  
 L2 cache/SRAM, 7-8  
 LBC, 14-8  
 PCI/PCI-X, 17-11  
 performance monitor, 20-3  
 security engine (SEC), 12-10  
 Memory space  
   PCI/PCI-X addressing, 17-47  
 Memory target queue  
   performance monitor events, 20-18  
 Message interrupts, *see* Interrupt controller (PIC), message  
   interrupts  
 MMU assist registers (MAS0–MAS4, MAS6–MAS7),  
   6-35–6-39  
 MMUCFG (MMU configuration register), *see* e500 core,  
   registers  
 MMUCSR0 (MMU control and status register 0), *see* e500  
   core, registers  
 MODT[0:3] (DDR on-die termination) signals, 9-8  
 MRAS (DDR row address strobe) signal, 9-7  
 MSR (machine state register), *see* e500 core, registers  
 MSRCID[0:4] (DDR/LBC debug source ID) signals, 4-20,  
   14-8, 21-3, 21-7  
 MWE (DDR write enable) signal, 9-8

## N

Nap mode, 1-20, 19-28  
*see also* Global utilities, power management

## O

On-chip memory  
   as L2 Cache, 1-13  
   as mapped SRAM, 1-13  
   overview, 1-12

## P

Page hit checking (LBC SDRAM), 14-50  
 Page management (LBC SDRAM), 14-50  
*PCI Express Base Specification, Rev. 1.0a*  
   *see* PCI Express controller  
 PCI Express controller  
   accessing configuration space  
     endpoint (EP) mode, 18-43  
     root complex (RC) mode, 18-42  
   address translation and mapping unit (ATMU)  
     inbound windows, 18-24  
       endpoint (EP) mode, 18-24  
       root complex (RC) mode, 18-25  
     outbound windows, 18-19  
   block diagram, 18-2  
   clocks  
     minimum CCB frequency equation, 4-22  
   commands  
     command register, 18-44  
   configuration space accesses, 18-42  
   error handling registers, 18-29–18-42  
   features, 18-3  
   latency timer, 18-49  
   modes of operation, 18-4  
     link width, 18-4  
     root complex or endpoint mode, 18-4  
   overview, 18-1  
   POR configuration, 18-4  
   power management, 18-13–18-18, 18-68–18-69  
   register descriptions  
     configuration header registers, 18-43–18-66  
       32-bit memory base address register, 18-51  
       64-bit high memory base address register, 18-52  
       64-bit low memory base address register, 18-52  
     base address registers, 18-50–18-52, 18-57  
     bridge control register, 18-66  
     bus status register, 18-46  
     cache line size register, 18-48  
     capabilities pointer register, 18-54, 18-64  
     command register, 18-44  
     configuration and status register base address  
       (PCSRBAR), 18-50, 18-57  
     device ID register, 18-44, 18-53  
     I/O base register, 18-59  
     I/O base upper 16 bits register, 18-63  
     I/O limit register, 18-59  
     I/O limit upper 16 bits register, 18-64  
     interrupt line register, 18-54, 18-65  
     interrupt pin register, 18-55, 18-65  
     latency timer register, 18-49, 18-59  
     maximum latency (EP-mode) register, 18-56  
     memory base register, 18-61

- memory limit register, 18-61
- minimum grant (EP-mode) register, 18-55
- prefetchable base upper 32 bits register, 18-63
- prefetchable limit upper 32 bits register, 18-63
- prefetchable memory base register, 18-62
- prefetchable memory limit register, 18-62
- primary bus number register, 18-57
- revision ID register, 18-47
- secondary bus number register, 18-58
- secondary status register, 18-60
- subordinate bus number register, 18-58
- subsystem vendor ID register, 18-53
- vendor ID register, 18-44
- device-specific configuration space registers, 18-67–18-80
  - capabilities register, 18-70
  - capability ID register, 18-70
  - data capabilities register, 18-71
  - device control register, 18-71
  - device status register, 18-72
  - link capabilities register, 18-73
  - link control register, 18-73
  - link status register, 18-74
  - MSI message address register, 18-79
  - MSI message capability ID register, 18-78
  - MSI message control register, 18-78
  - MSI message data register, 18-80
  - MSI message upper address register, 18-79
  - power management capabilities register, 18-68
  - power management capability ID register, 18-68
  - power management data register, 18-69
  - power management status and control register, 18-69
  - root control register, 18-77
  - root status register, 18-78
  - slot capabilities register, 18-75
  - slot control register, 18-76
  - slot status register, 18-76
- extended configuration space registers, 18-81–18-89
  - advanced error capabilities and control register, 18-86
  - advanced error reporting capability ID register, 18-82
  - correctable error mask register, 18-85
  - correctable error source ID register, 18-89
  - correctable error status register, 18-85
  - error source ID register, 18-89
  - header log register, 18-87
  - root error command register, 18-88
  - root error status register, 18-88
  - uncorrectable error mask register, 18-83
  - uncorrectable error severity register, 18-84
  - uncorrectable error status register, 18-82
- memory-mapped registers, 18-5
  - ATMU registers, 18-19–18-29
    - by acronym, *see* Register Index
    - configuration access registers, 18-9–18-12, 18-42–18-43
    - error management registers, 18-29–18-42
    - IP block revision registers, 18-18–18-19
    - pwr mgmt and message registers, 18-13–18-18
    - signals summary, 18-4
      - see also* Signals, PCI Express
- PCI Local Bus Specification configuration registers
  - see* PCI/PCI-X controller, registers
- PCI/PCI-X controller
  - 64-bit/32-bit bus, 17-5
  - address bus decoding, 17-47
  - address translation and mapping unit (ATMU)
    - inbound windows (4), 2-10, 17-19
    - outbound windows (4), 17-16
  - arbiter configuration (POR), 4-20, 17-5
  - block diagram, 17-2
  - burst operations
    - cache wrap mode, 17-47
    - linear incrementing, 17-47
  - bus arbitration, 17-5, 17-42
  - bus protocol, 17-45
    - burst operation, 17-45
    - command encodings, 17-46
  - clocking, 17-45, 17-50
  - commands
    - command register, 17-31, 17-59
    - encodings, 17-46
    - interrupt-acknowledge transactions, 17-63
    - special-cycle, 17-64
  - configuration cycles, 17-58
  - configuration space addressing, 17-48
  - data bus width (POR), 4-19
  - debug mode
    - source and target ID (PCI\_AD[63:59]), 21-25
  - error handling, 17-65–17-66
    - address/data parity, 17-54, 17-65, 17-66
    - detection and reporting, 17-65
    - reporting
      - $\overline{\text{PERR}}$  and  $\overline{\text{SERR}}$  signals, 17-66
      - target-initiated termination, 17-53
    - retry transactions, 17-53
    - target-abort, 17-53
    - target-disconnect, 17-53
  - features, 17-4
  - functional description, 17-42
  - I/O impedance (POR), 4-20
  - I/O space addressing, 17-48
  - initialization/application information, 17-67–17-70
  - interrupts
    - error enable register, 17-26

- latency timer, 17-36, 17-54, 17-59
- memory map/register definition, 17-11
- memory space addressing, 17-47
- modes of operation, 17-4
  - agent configuration lock mode, 17-68
  - agent mode, 17-68
  - cache wrap mode, 17-47
  - host mode, 17-68
  - linear incrementing, 17-47
  - PCI-X mode
    - selection (POR), 4-19
- overview, 17-2
- performance monitor events
  - common events, 20-21
- POR configuration, 17-67
- power management
  - special-cycle operations, 17-64
- register descriptions
  - configuration header registers, 17-30–17-42, 17-59
    - 32-bit memory base address register, 17-37
    - 64-bit high memory base address register, 17-38
    - 64-bit low memory base address register, 17-37
  - arbiter configuration register (PBACR), 17-42
  - base address registers, 17-36–17-38
  - base class code register, 17-35
  - bus function register (PBFR), 17-41
  - bus status register, 17-32, 17-49, 17-53, 17-65, 17-66
  - cache line size register, 17-35
  - capabilities pointer register, 17-39
  - command register, 17-31, 17-59
  - configuration and status register base address (PCSRBAR), 17-36
  - device ID register, 17-31, 17-39
  - interrupt line register, 17-39
  - interrupt pin register, 17-40
  - latency timer register, 17-36
  - maximum grant (MAX GNT) register, 17-40
  - maximum latency (MAX LAT) register, 17-41
  - programming interface register, 17-34
  - revision ID register, 17-34
  - subclass code register, 17-35
  - vendor ID register, 17-30, 17-38
- memory-mapped registers, 17-11
  - ATMU inbound registers, 17-19–17-23
  - ATMU outbound registers, 17-16–17-19
  - by acronym, *see* Register Index
  - configuration access registers, 17-14–17-15, 17-60
  - error management registers, 17-23–17-29
- signals summary, 17-6
  - see also* Signals, PCI/PCI-X
- target-abort termination, 17-53
- target-disconnect cycles, 17-4, 17-53
- target-initiated termination
  - target-abort error, 17-53
  - target-disconnect, 17-4, 17-53
- transactions
  - fast back-to-back transactions, 17-56
  - interrupt-acknowledge transactions, 17-63
  - read transactions, 17-50
  - retry transactions, 17-53
  - special-cycle transactions, 17-64
  - timing diagrams, 17-50
  - transaction termination, 17-52
    - bus status register, termination status, 17-54
    - completion, 17-53
    - master-abort termination, 17-53
    - master-initiated, 17-52
    - target-initiated, 17-53, 17-54
    - timeout, 17-53
    - write transactions, 17-50, 17-51
  - turnaround cycle, 17-49
- PCI\_AD[47:40] signals as GP I/O, *see* Global utilities, general-purpose I/O signals
- PCI\_AD[63:0] (PCI address/data bus) signals, 17-6
- PCI\_C/BE[7:0] (PCI command/byte enable) signals, 17-7, 17-46, 17-48, 17-49, 17-65
- PCI\_DEVSEL (PCI device select) signal, 17-7, 17-48
- PCI\_FRAME (PCI frame) signal, 17-7, 17-45
- PCI\_GNT[4:0] (PCI bus grant) signals, 17-8, 17-43
- PCI\_IDSEL (PCI initialization device) signal, 17-8
- PCI\_IRDY (PCI initiator ready) signal, 17-8, 17-45
- PCI\_PAR (PCI parity) signal, 17-9
- PCI\_PERR (PCI parity error) signal, 17-9, 17-66
- PCI\_REQ[4:0] (PCI bus request) signals, 17-9, 17-42
- PCI\_SERR (PCI system error) signal, 17-10, 17-66
- PCI\_STOP (PCI stop) signal, 17-10, 17-49
- PCI\_TRDY (PCI target ready) signal, 17-10, 17-45, 17-52
- Performance monitor (device)
  - block diagram, 20-2
  - burstiness, 20-14, 20-28
  - control registers, 20-5–20-10
  - counters (PMCn)
    - chaining, 20-13
    - registers, 20-10
    - triggering, 20-13
  - event counting, 20-12
  - events, 20-16–20-27
    - chaining, 20-27
    - DDR controller, 20-17
    - debug, 20-27
    - DMA controller, 20-18
    - DUART, 20-27
    - e500 coherency module (ECM), 20-19
    - interrupt controller (PIC), 20-20

- L2 cache/SRAM, 20-26
  - local bus controller (LBC), 20-26
  - memory target queue, 20-18
  - PCI/PCI-X common events, 20-21
  - events triggered by watchpoint monitor, 21-27
  - examples, 20-28
    - burstiness event, 20-14
    - burstiness event counting, 20-28
    - simple event counting, 20-28
    - threshold event counting, 20-28
    - triggering event counting, 20-28
  - external signals, 20-3
  - features, 20-3
  - functional description, 20-11
  - interrupts, 20-11
  - interrupts (from PIC) to generate events, 10-31
  - masking interrupts (from PIC), 10-31
  - memory map/register definition, 20-3
  - overflow indication on TRIG\_OUT, 21-25
  - overview, 20-2
  - threshold events, 20-12
  - Phase-locked loops (PLLs)
    - POR status (global utilities), 19-4
  - PIDn (process ID registers 0–2), *see* e500 core, registers
  - PIR (processor ID register), *see* e500 core, registers
  - PMCn (performance monitor counter registers 0–3), *see* e500 core, registers
  - PMGC0 (performance monitor global control register 0), *see* e500 core, registers
  - PMLCan (performance monitor local control registers a0–a3), *see* e500 core, registers
  - PMLCbn (performance monitor local control registers b0–b3), *see* e500 core, registers
  - Power management
    - block disable
      - block disable control (DEVDISR), 19-14, 19-27
      - LBC, 14-4
    - DDR interface, 9-61
    - device low-power modes, 19-25–19-32
      - control and status register (POWMGTCSR), 19-16
      - READY negation, 4-2
    - interrupts that cause wake-up, 10-3
    - overview, 1-20
    - PCI Express, 18-13–18-18, 18-68–18-69
    - PCI special-cycle operations, 17-64
    - see also* Global utilities, power management
  - Power-on reset (POR)
    - configuration
      - boot ROM location, 4-13
      - boot sequencer configuration, 4-16
    - clock
      - e500 core PLL ratio, 4-12
      - system/CCB PLL ratio, 4-11
    - CPU boot configuration, 4-15
    - DDR debug mode (ECC pins used for debug), 4-20, 21-3
    - eTSEC1 protocol, 4-18
    - eTSEC1–2 data width, 4-17
    - eTSEC3 protocol, 4-18
    - general-purpose (external system)
      - configuration—LAD[0:31] (GPPORCR), 4-21
    - memory debug select (DDR or LBC), 4-20, 21-3
    - PCI data bus width, 4-19
    - PCI debug configuration, 21-3
    - PCI I/O impedance, 4-20
    - PCI/PCI-X arbiter configuration, 4-20
    - PCI/PCI-X, modes of operation, 17-67
    - PCI-X mode selection, 4-19
    - configuration reporting
      - global utilities, 19-4, 19-6, 19-7, 19-8, 19-10, 19-11
    - debug modes summary, 21-3
    - hard reset, 4-8
    - output signal states during reset, 3-17
    - PCI Express, modes of operation, 18-4
    - reset configuration signals, 3-15
    - sequence of events, 4-9
      - and READY signal, 4-2, 4-10
  - Processor version (PVR), 19-18
  - Programmable interrupt controller (PIC), *see* Interrupt controller (PIC), 1-15
  - Protocols
    - PCI, *see* PCI/PCI-X controller, bus protocol
  - PVR (processor version register), *see* e500 core, registers
- ## Q
- Quality of service (QoS), *see* eTSEC
- ## R
- Random number generator (RNG), *see* Security engine (SEC)
  - RapidIO controller
    - clocks
      - minimum CCB frequency equation, 4-22
      - operation, 4-22
  - READY signal, 4-2, 4-10, 21-24, 21-25
  - Registers
    - by acronym (memory-mapped registers)
      - see* Register Index
    - configuration, control, and status, 2-10–2-14, 4-4
    - device-specific utilities, 2-13
    - general utilities, 2-12
    - programmable interrupt controller (PIC) space, 2-13
    - context ID, 21-23–21-24
    - DDR

- configuration registers, 9-11–9-32
- error handling registers, 9-34–9-41
- error injection registers, 9-33–9-34
- e500 core, see e500 core, registers
- ECM, 8-3
- eTSEC, 15-22–15-126
  - DMA attribute registers, 15-110–15-112
  - FIFO registers, 15-109–15-110
  - general control and status registers, 15-22–15-36
  - hash function registers, 15-107–15-108
  - lossless flow control registers, 15-112–15-114
  - MAC registers, 15-66–15-79
  - MIB registers, 15-79–15-107
  - receive control and status registers, 15-48–15-63
  - ten-bit interface registers, 15-115–15-126
  - transmit control and status registers, 15-36–15-48
- global utilities, 19-4
  - POR boot mode status, 19-6
  - POR debug mode status, 19-10
  - POR device status, 19-8
  - POR external system configuration, 19-11
  - POR I/O impedance status, 19-7
  - POR PLL status, 19-4
- I<sup>2</sup>C interface, 11-5
- L2 cache/SRAM registers, 7-8–7-25
- LBC, 14-10
- local access window registers
  - attributes registers (LAWAR0–LAWAR7), 2-7
  - base address registers (LAWBAR0–LAWBAR7), 2-7
- PCI Express
  - configuration header registers, 18-43–18-66
    - 32-bit memory base address register, 18-51
    - 64-bit high memory base address register, 18-52
    - 64-bit low memory base address register, 18-52
  - base address registers, 18-50–18-52, 18-57
  - bridge control register, 18-66
  - bus status register, 18-46
  - cache line size register, 18-48
  - capabilities pointer register, 18-54, 18-64
  - command register, 18-44
  - configuration and status register base address (PCSRBAR), 18-50, 18-57
  - device ID register, 18-44, 18-53
  - I/O base register, 18-59
  - I/O base upper 16 bits register, 18-63
  - I/O limit register, 18-59
  - I/O limit upper 16 bits register, 18-64
  - interrupt line register, 18-54, 18-65
  - interrupt pin register, 18-55, 18-65
  - latency timer register, 18-49, 18-59
  - maximum latency (EP-mode) register, 18-56
  - memory base register, 18-61
  - memory limit register, 18-61
  - minimum grant (EP-mode) register, 18-55
  - prefetchable base upper 32 bits register, 18-63
  - prefetchable limit upper 32 bits register, 18-63
  - prefetchable memory base register, 18-62
  - prefetchable memory limit register, 18-62
  - primary bus number, 18-57
  - revision ID register, 18-47
  - secondary bus number, 18-58
  - secondary status register, 18-60
  - subordinate bus number, 18-58
  - subsystem vendor ID, 18-53
  - vendor ID, 18-44
- device-specific configuration space registers, 18-67–18-80
  - capabilities register, 18-70
  - capability ID register, 18-70
  - device capabilities register, 18-71
  - device control register, 18-71
  - device status register, 18-72
  - link capabilities register, 18-73
  - link control register, 18-73
  - link status register, 18-74
  - MSI message address register, 18-79
  - MSI message capability ID register, 18-78
  - MSI message control register, 18-78
  - MSI message data register, 18-80
  - MSI message upper address register, 18-79
  - power management capabilities register, 18-68
  - power management capability ID register, 18-68
  - power management data register, 18-69
  - power management status and control register, 18-69
  - root control register, 18-77
  - root status register, 18-78
  - slot capabilities register, 18-75
  - slot control register, 18-76
  - slot status register, 18-76
- extended configuration space registers, 18-81–18-89
  - advanced error capabilities and control register, 18-86
  - advanced error reporting capability ID register, 18-82
  - correctable error mask register, 18-85
  - correctable error source ID register, 18-89
  - correctable error status register, 18-85
  - error source ID register, 18-89
  - header log register, 18-87
  - root error command register, 18-88
  - root error status register, 18-88
  - uncorrectable error mask register, 18-83
  - uncorrectable error severity register, 18-84
  - uncorrectable error status register, 18-82
- memory-mapped registers
  - ATMU registers, 18-19–18-29



- configuration access registers, 18-9–18-12, 18-42–18-43
  - error management registers, 18-29–18-42
  - IP block revision registers, 18-18–18-19
  - pwr mgmt and message registers, 18-13–18-18
  - PCI/PCI-X
    - configuration header registers, 17-30–17-42, 17-59
    - 32-bit memory base address register, 17-37
    - 64-bit high memory base address register, 17-38
    - 64-bit low memory base address register, 17-37
    - arbiter configuration register (PBACR), 17-42
    - base address registers, 17-36–17-38
    - base class code register, 17-35
    - bus function register (PBFR), 17-41
    - bus status register, 17-32, 17-49, 17-53, 17-65, 17-66
    - cache line size register, 17-35
    - capabilities pointer register, 17-39
    - command register, 17-31, 17-59
    - configuration and status register base address (PCSRBAR), 17-36
    - device ID register, 17-31, 17-39
    - interrupt line register, 17-39
    - interrupt pin register, 17-40
    - latency timer register, 17-36
    - maximum grant (MAX GNT) register, 17-40
    - maximum latency (MAX LAT) register, 17-41
    - programming interface register, 17-34
    - revision ID register, 17-34
    - subclass code register, 17-35
    - vendor ID, 17-30, 17-38
  - memory-mapped registers
    - ATMU inbound registers, 17-19–17-23
    - ATMU outbound registers, 17-16–17-19
    - configuration access registers, 17-14–17-15, 17-60
    - error management registers, 17-23–17-29
  - performance monitor, descriptions, 20-3
  - PIC, 10-17
    - global registers, 10-17–10-22
    - global timer registers, 10-22–10-27
    - interrupt source configuration registers, 10-22–10-25, 10-38–10-43
    - message registers, 10-33–10-35
    - non-accessible registers
      - in-service register (ISR), 10-49
      - interrupt pending register (IPR), 10-47
      - interrupt request register (IRR), 10-47
    - per-CPU registers, 10-43–10-47
    - performance monitor mask registers, 10-31–10-33
    - summary registers, 10-27–10-31
  - processor version register (PVR), 19-18
  - security engine (SEC)
    - AESU, 12-67–12-79
    - AFEU, 12-42–12-50
    - controller registers, 12-109–12-115
    - crypto-channel, 12-92–12-102
    - DEU, 12-33–12-41
    - interrupt registers, 12-110–12-112
    - KEU, 12-80–12-91
    - MDEU, 12-50–12-62
    - PKEU, 12-26–12-32
    - RNG, 12-63–12-67
    - system version register (SVR), 19-19
    - trace buffer, 21-16–21-23
    - trigger out source register, 21-24
    - watchpoint monitor, 21-10–21-16
  - Reset
    - core reset through PIC register, 10-20
    - hard reset actions, 4-8
    - HRESET\_REQ control, 19-20
    - operations, 4-8
    - power-on reset (POR)
      - configuration, see Power-on reset (POR), configuration sequence of events, 4-9
      - requests from RapidIO and PCI Express, 19-18
    - SEC, channel reset, 12-103
    - signals summary, 4-2
      - see also Signals, reset
    - soft request, 19-18
    - soft reset actions, 4-8
      - and reconfiguring the eTSEC, 15-142
  - RMON support, see eTSEC, modes of operation
  - RTC (real time clock) signal, 4-3, 4-23, 10-25, 10-26, 19-29
  - RTS, see DUART\_RTS[0:1]
- ## S
- SCL (I<sup>2</sup>C serial clock) signal, 11-3, 11-4
  - SD\_RX[7:0]/SD\_RX̄[7:0] (PCI Express serial data input and complement) signals, 18-5
  - SD\_TX[7:0]/SD\_TX̄[7:0] (PCI Express serial data output and complement) signals, 18-5
  - SDA (I<sup>2</sup>C serial data) signal, 11-3, 11-4
  - SDRAM interface (LBC), 14-47–14-58
    - <italics>see also Local bus controller (LBC), SDRAM interface
  - Security engine (SEC)
    - advanced encryption standard execution unit (AESU), 12-7, 12-67
    - contexts
      - CBC mode, 12-76
      - CCM mode, 12-77
      - counter mode, 12-76
      - SRT mode, 12-76
    - restore decrypt key (RDK) operation, 12-69
    - Rijndael algorithm, 12-7

- ARC Four execution unit (AFEU), 12-6
  - context, 12-49
  - dump context mode, 12-42
  - host-provided context via prevent permute, 12-42
- block diagram, 12-3
- data encryption standard execution unit (DEU), 12-6, 12-33
- descriptors, 12-4, 12-16
  - header dword, 12-17
  - descriptor types, 12-19
  - EU selection, 12-18
  - writeback format, 12-94
  - pointer dwords 0–6, 12-20
  - descriptor types and formats, 12-24
- Diffie-Hellman key exchanges, 12-6
- disabling the block, 12-115
- ECC digital signatures, 12-6
- error handling, 12-98, 12-103
  - see also* individual execution units
- execution units (EUs), 12-25
  - channels, 12-91
    - arbitration of internal buses, 12-105
    - assignment of EUs to channels, 12-104
    - crypto-channel states, 12-98
    - priority arbitration, 12-104, 12-105
    - controller arbitration, 12-106
- features, 12-1
- interrupts, 12-103–12-104, 12-108
  - registers, 12-110–12-112
- Kasumi execution unit (KEU), 12-8, 12-79
- link tables, 12-102
  - format, 12-21
  - gather operation (reading data), 12-17, 12-97
  - scatter operation (writing data), 12-17, 12-97
- memory map/register definition, 12-10
- message digest execution unit (MDEU), 12-7, 12-50
  - algorithm selection, 12-52
  - endian (byte ordering) considerations, 12-60, 12-61, 12-62
  - recommended settings for mode register, 12-53
- overview
  - architecture, 12-2
  - execution units (EUs), 12-5
  - security channels, 12-8
  - security controller access types, 12-9
- public key execution unit (PKEU), 12-5, 12-26
  - Chinese remainder theorem and RSA algorithm, 12-6
  - elliptic curve operations, 12-5
  - modular exponentiation operations, 12-6
  - Montgomery modular multiplication algorithm, 12-6
  - parameter memories A, B, E, and N, 12-32
- random number generator (RNG), 12-7, 12-62

- register descriptions
  - AESU, 12-67–12-79
  - AFEU, 12-42–12-50
  - by acronym, *see* Register Index
  - controller registers, 12-109–12-115
  - crypto-channel, 12-92–12-102
  - DEU, 12-33–12-41
  - interrupt registers, 12-110–12-112
  - KEU, 12-80–12-91
  - MDEU, 12-50–12-62
  - PKEU, 12-26–12-32
  - RNG, 12-63–12-67
- reset channel, 12-103
- transactions (internal)
  - master reads, 12-107
  - master writes, 12-107
  - slave reads and writes, 12-107
- Serial data/clock, *see* I<sup>2</sup>C interface, 11-1
- Signals
  - clock
    - RTC (real time clock), 4-3, 4-23, 10-25, 10-26, 19-29
    - SYCLK (system clock input), 4-3
  - complete signal listing
    - alphabetical reference, 3-10
    - configuration signals, sampled at POR, 3-15
      - see also* Power-on reset (POR)
    - figure showing groupings, 3-1
    - output signal states at power-on reset, 3-17
    - reference by functional block, 3-5
- DDR
  - MA[0:14] (address bus), 9-7
  - MBA[0:1] (logical bank address), 9-7
  - MCAS (column address strobe), 9-7
  - MCK[0:5] (DDR clock output complements), 9-9
  - MCK[0:5] (DDR clock outputs), 9-9
  - MCKE[0:3] (DDR clock enables), 9-9
  - MCS[0:3] (chip selects), 9-8
  - MDIC[0:1] (driver impedance calibration), 9-8
  - MDM[0:8] (SDRAM data output mask), 9-8
  - MDQS[0:8] (data bus strobes), 9-6, 9-42
  - MDVAL (debug mode data valid), 4-20, 21-3, 21-6
  - MECC[0:5] (error correcting code)
    - as debug signals, 21-3, 21-7
  - MECC[0:7] (error correcting code), 4-20, 9-6
  - MODT[0:3] (on-die termination), 9-8
  - MRAS (row address strobe), 9-7
  - MSRCID[0:4] (debug source ID), 4-20, 21-3, 21-7
  - MWE (write enable), 9-8
- DMA
  - DMA\_DACK[0:3] (DMA acknowledge), 16-6
  - DMA\_DDONE[0:3] (DMA done), 16-6
  - DMA\_DREQ[0:3] (DMA request), 16-6

- DUART
  - UART\_CTS[0:1] (DUART clear to send), 13-1, 13-3
  - UART\_RTS[0:1] (DUART request to send), 13-1, 13-3, 13-4
  - UART\_SIN [0:1] (DUART transmitter serial data in), 13-2, 13-3
  - UART\_SOUT [0:1] (DUART transmitter serial data out), 13-2, 13-3
- eTSEC
  - EC\_GTX\_CLK125 (eTSEC gigabit transmit 125 MHz source), 15-10
  - EC\_MDC (eTSEC management data clock), 15-10
  - EC\_MDIO (eTSEC management data input/output, BIDI), 15-10
  - FIFO interface signal summary, 15-140
  - TSECn\_COL (eTSEC 1–4 collision input), 15-9
  - TSECn\_CRS (eTSEC 1–4 carrier sense input/FIFO receiver flow control), 15-9
  - TSECn\_GTX\_CLK (eTSEC 1–4 gigabit transmit clock), 15-9
  - TSECn\_RX\_CLK (eTSEC 1–4 receive clock), 15-10
  - TSECn\_RX\_DV (eTSEC 1–4 receive data valid), 15-10
  - TSECn\_RX\_ER (eTSEC 1–4 receive error), 15-11
  - TSECn\_RXD[7:0] (eTSEC 1–4 receive data in), 15-11
  - TSECn\_TX\_CLK (eTSEC 1–4 transmit clock in), 15-11
  - TSECn\_TX\_EN (eTSEC 1–4 transmit data valid), 15-12
  - TSECn\_TX\_ER (eTSEC 1–4 transmit error), 15-12
  - TSECn\_TXD[7:0] (eTSEC 1–4 transmit data out), 15-12
- global utilities
  - ASLEEP, 19-2, 19-28
  - CKSTP\_IN (checkstop in), 19-2
  - CKSTP\_OUT (checkstop out), 19-3
  - CLK\_OUT, 19-3, 19-23
  - GPOUT[24:31], 19-3
- I<sup>2</sup>C
  - SCL (serial clock), 11-3, 11-4
  - SDA (serial data), 11-3, 11-4
- JTAG
  - TCK (JTAG test clock), 21-8
  - TDI (JTAG test data input), 21-8
  - TDO (JTAG test data output), 21-8
  - TMS (JTAG test mode select), 21-8
  - TRST (JTAG test reset), 21-9
- LBC
  - LA[27:31] (non-multiplexed address), 14-7
  - LAD[0:31] (multiplexed address/data), 14-7
  - LALE (external address latch enable), 14-5, 14-33
  - LBCTL (data buffer control), 14-7, 14-35
  - LBS[0:3] (UPM byte select), 14-6
  - LCK[0:2] (clock), 14-8
  - LCKE (clock enable), 14-7
  - LCS[0:7] (chip select), 14-5
  - LCS0 (LBC chip select 0), 14-46
  - LDP[0:3] (data parity), 14-7, 14-36
  - LGPL0 (GP line 0), 14-6
  - LGPL1 (GP line 1), 14-6
  - LGPL2 (GP line 2), 14-6
  - LGPL3 (GP line 3), 14-6
  - LGPL4 (GP line 4), 14-6
  - LGPL5 (GP line 5), 14-7
  - LGTA (GPCM transfer acknowledge), 14-6, 14-46
  - LOE (GPCM output enable), 14-6
  - LPBSE (parity byte select), 14-6
  - LSDA10 (SDRAM A10), 14-6
  - LSDCAS (SDRAM CAS), 14-6
  - LSDDQM[0:3] (SDRAM data mask), 14-6
  - LSDRAS (SDRAM RAS), 14-6
  - LSDWE (SDRAM write enable), 14-6
  - LSYNC\_IN (PLL synchronization in), 14-8
  - LSYNC\_OUT (PLL synchronization out), 14-8
  - LWE[0:3] (GPCM write enable), 14-6
  - MDVAL (debug mode data valid), 4-20, 14-8, 21-3, 21-6
  - MSRCID[0:4] (debug source ID), 4-20, 14-8, 21-3, 21-7
  - TA (data transfer acknowledge), 14-34
  - UPWAIT (UPM wait), 14-6, 14-59
- other
  - TEST\_SEL (factory test), 21-6
  - THERM[0:1] (thermal resistor access), 21-9
- PCI Express
  - SD\_RX[7:0]/SD\_RX[7:0] (PCI Express serial data input and complement) signals, 18-5
  - SD\_TX[7:0]/SD\_TX[7:0] (PCI Express serial data output and complement) signals, 18-5
- PCI/PCI-X
  - PCI\_AD[63:0] (address/data bus), 17-6
  - PCI\_C/BE[7:0] (command/byte enable), 17-7, 17-46, 17-48, 17-49, 17-65
  - PCI\_DEVSEL (device select), 17-7, 17-48
  - PCI\_FRAME (frame), 17-7, 17-45
  - PCI\_GNT[4:0] (bus grant), 17-8, 17-43
  - PCI\_IDSEL (initialization device), 17-8
  - PCI\_IRDY (initiator ready), 17-8, 17-45
  - PCI\_PAR (parity), 17-9
  - PCI\_PERR (parity error), 17-9, 17-66
  - PCI\_REQ[4:0] (bus request), 17-9, 17-42
  - PCI\_SERR (system error), 17-10, 17-66
  - PCI\_STOP (stop), 17-10, 17-49
  - PCI\_TRDY (target ready), 17-10, 17-45
- PIC
  - IRQ[0:11], 10-8
  - IRQ\_OUT, 10-8, 10-27
  - MCP, 10-8
  - UDE, 10-8
- reset

HRESET (hard reset), 4-2, 4-8  
 HRESET\_REQ (hard reset request), 4-2, 11-18, 11-19  
 READY, 4-2, 21-24, 21-25  
 SRESET (soft reset), 4-2, 4-8  
 watchpoint monitor  
   TRIG\_IN (watchpoint trigger in), 21-7, 21-11, 21-18  
   TRIG\_OUT (watchpoint trigger out), 21-8, 21-24  
 Sleep mode, 1-20, 19-28, 19-32  
   *see also* Global utilities, power management  
 Snooping  
   power management and snooping (global utilities), 19-32  
 Soft reset, 19-18  
 SPEFSCR (signal processing and embedded floating-point status and control register), *see* e500 core, registers  
 SPRGn (software-use registers 0–7), *see* e500 core, registers  
 SRAM, *see* L2 cache/SRAM, 7-27  
 SRESET (soft reset) signal, 4-2, 4-8  
 SRR0–1 (save/restore registers 0–1), *see* e500 core, registers  
 Stashing, *see* L2 cache/SRAM, stashing, 7-25  
 SVR (system version register), *see* e500 core, registers  
 SYSCLK (system clock input) signal, 4-3

## T

TA (LBC data transfer acknowledge) signal, 14-34  
 Target-disconnect, *see* PCI/PCI-X controller  
 TBL (time base lower register), *see* e500 core, registers  
 TBU (time base upper register), *see* e500 core, registers  
 TCK (JTAG test clock) signal, 21-8  
 TCR (timer control register), *see* e500 core, registers  
 TDI (JTAG test data input) signal, 21-8  
 TDO (JTAG test data output) signal, 21-8  
 Termination  
   PCI/PCI-X, termination of PCI transactions, 17-52  
 Test interface, *see* JTAG test access port  
 TEST\_SEL (factory test) signal, 21-6  
 THERM[0:1] (thermal resistor access) signals, 21-9  
 Timing diagrams  
   PCI/PCI-X transactions, 17-50  
 TLB0CFG (TLB0 configuration register), *see* e500 core, registers  
 TLB1CFG (TLB1 configuration register), *see* e500 core, registers  
 TMS (JTAG test mode select) signal, 21-8  
 Trace buffer  
   and watchpoint monitor, block diagram, 21-1  
   as a second watchpoint monitor, 21-28  
   functional description, 21-28–21-31  
   initialization, 21-31  
   modes of triggering and arming, 21-4  
   overview, 21-1  
   register descriptions, 21-16–21-23  
   by acronym, *see* Register Index

*see also* Watchpoint monitor, 21-4  
   traced data formats relative to TBCR1[IFSEL]  
     DDR trace buffer entry, 21-29  
     ECM trace buffer entry, 21-28  
     PCI trace buffer entry, 21-30  
 Transactions  
   PCI/PCI-X *see* PCI/PCI-X controller, transactions  
 TRIG\_IN (watchpoint trigger in) signal, 21-7, 21-11, 21-18  
 TRIG\_OUT (watchpoint trigger out) signal, 21-8, 21-24  
 TRST (JTAG test reset) signal, 21-9  
 TSECn\_COL (eTSEC 1–4 collision input) signals, 15-9  
 TSECn\_CRS (eTSEC 1–4 carrier sense input/FIFO receiver flow control) signals, 15-9  
 TSECn\_GTX\_CLK (eTSEC 1–4 gigabit transmit clock) signals, 15-9  
 TSECn\_RX\_CLK (eTSEC 1–4 receive clock) signals, 15-10  
 TSECn\_RX\_DV (eTSEC 1–4 receive data valid) signals, 15-10  
 TSECn\_RX\_ER (eTSEC 1–4 receive error) signals, 15-11  
 TSECn\_RXD[7:0] (eTSEC 1–4 receive data in) signals, 15-11  
 TSECn\_TX\_CLK (eTSEC 1–4 transmit clock in) signals, 15-11  
 TSECn\_TX\_EN (eTSEC 1–4 transmit data valid) signals, 15-12  
 TSECn\_TX\_ER (eTSEC 1–4 transmit error) signals, 15-12  
 TSECn\_TXD[7:0] (eTSEC 1–4 transmit data out) signals, 15-12  
 TSR (timer status register), *see* e500 core, registers

## U

UART\_CTS[0:1] (DUART clear to send) signals, 13-1, 13-3  
 UART\_RTS[0:1] (DUART request to send) signals, 13-1, 13-3, 13-4  
 UART\_SIN [0:1] (DUART transmitter serial data in) signals, 13-2, 13-3  
 UART\_SOUT [0:1] (DUART transmitter serial data out) signals, 13-2, 13-3  
 UDE (unconditional debug event) signal, 10-8  
 Universal asynchronous receiver/transmitter, *see* DUART  
 UPMCn (user performance monitor counter registers 0–3), *see* e500 core, registers  
 UPMGC0 (user performance monitor global control register 0), *see* e500 core, registers  
 UPMLCan (user performance monitor local control registers a0–a3), *see* e500 core, registers  
 UPMLCbn (user performance monitor local control registers b0–b3), *see* e500 core, registers  
 UPWAIT (LBC UPM wait) signal, 14-6, 14-59  
 USPRG0 (user software-use register 0), *see* e500 core, registers

**V**

Voltage selection  
LBC signals, 19-20

**W**

Watchpoint monitor  
and trace buffer, block diagram, 21-1  
functional description, 21-27  
initialization, 21-31  
modes of triggering and arming, 21-4  
overview, 21-1  
performance monitor events, 21-27  
register descriptions, 21-10–21-16  
by acronym, see Register Index  
second WM by using trace buffer, 21-28  
see also Trace buffer, 21-4  
signals summary, 21-5  
see also Signals, watchpoint, 21-5

**X**

XER (integer exception register), see e500 core, registers

**Z**

ZBT SRAM interface (LBC), 14-96



<b>Part I—Overview</b>	<b>I</b>
Overview	<b>1</b>
Memory Map	<b>2</b>
Signal Descriptions	<b>3</b>
Reset, Clocking, and Initialization	<b>4</b>
<b>Part II—e500 Core Complex and L2 Cache</b>	<b>II</b>
Core Complex Overview	<b>5</b>
Core Register Summary	<b>6</b>
L2 Look-Aside Cache/SRAM	<b>7</b>
<b>Part III—Memory, Security, and I/O Interfaces</b>	<b>III</b>
e500 Coherency Module	<b>8</b>
DDR Memory Controller	<b>9</b>
Programmable Interrupt Controller	<b>10</b>
I <sup>2</sup> C Interfaces	<b>11</b>
Security Engine (SEC) 2.1	<b>12</b>
DUART	<b>13</b>
Local Bus Controller	<b>14</b>
Enhanced Three-Speed Ethernet Controllers	<b>15</b>
DMA Controller	<b>16</b>
PCI Bus Interface	<b>17</b>
PCI Express Interface Controller	<b>18</b>
<b>Part IV—Global Functions and Debug</b>	<b>IV</b>
Global Utilities	<b>19</b>
Device Performance Monitor	<b>20</b>
Debug Features and Watchpoint Facility	<b>21</b>
Revision History	<b>A</b>
Complete List of Configuration, Control, and Status Registers	<b>B</b>
Glossary	<b>GLO</b>
Index	<b>IND</b>

<b>I</b>	<b>Part I—Overview</b>
<b>1</b>	Overview
<b>2</b>	Memory Map
<b>3</b>	Signal Descriptions
<b>4</b>	Reset, Clocking, and Initialization
<b>II</b>	<b>Part II—e500 Core Complex and L2 Cache</b>
<b>5</b>	Core Complex Overview
<b>6</b>	Core Register Summary
<b>7</b>	L2 Look-Aside Cache/SRAM
<b>III</b>	<b>Part III—Memory, Security, and I/O Interfaces</b>
<b>8</b>	e500 Coherency Module
<b>9</b>	DDR Memory Controller
<b>10</b>	Programmable Interrupt Controller
<b>11</b>	I <sup>2</sup> C Interfaces
<b>12</b>	Security Engine (SEC) 2.1
<b>13</b>	DUART
<b>14</b>	Local Bus Controller
<b>15</b>	Enhanced Three-Speed Ethernet Controllers
<b>16</b>	DMA Controller
<b>17</b>	PCI Bus Interface
<b>18</b>	PCI Express Interface Controller
<b>IV</b>	<b>Part IV—Global Functions and Debug</b>
<b>19</b>	Global Utilities
<b>20</b>	Device Performance Monitor
<b>21</b>	Debug Features and Watchpoint Facility
<b>A</b>	Revision History
<b>B</b>	Complete List of Configuration, Control, and Status Registers
<b>GLO</b>	Glossary
<b>IND</b>	Index